UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR


TRABAJO DE FIN DE MÁSTER

# DESARROLLO, ESTUDIO Y ANÁLISIS DE UNA PLATAFORMA DE NARIZ ELECTRÓNICA EN LAZO CERRADO

Máster Universitario en Ingeniería de Telecomunicación

Autor: Carlos Mario Velasco Botina
Tutor: Francisco de Borja Rodríguez Ortíz

FECHA: JULIO 2018

# DESARROLLO, ESTUDIO Y ANÁLISIS DE UNA PLATAFORMA DE NARIZ ELECTRÓNICA EN LAZO CERRADO

AUTOR: Carlos Mario Velasco Botina
DIRECTOR: Francisco de Borja Rodríguez Ortíz

Grupo de Neurocomputación Biológica
Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
JULIO 2018

# Resumen


## Resumen

En este trabajo se plantea la construcción de una plataforma de experimentación de nariz electrónica en lazo cerrado, la cual será usada para la adquisición y discriminación de distintos odorantes bajo estudio. El desarrollo de las narices electrónicas actualmente es amplio y desde hace unos años este campo de estudio ha venido creciendo de forma gradual siendo un campo activo de investigación en la actualidad. La construcción de esta plataforma de experimentación plantea el reto de que esta debe ser versátil para el uso de distintas técnicas de modulación implementadas previamente en otros trabajos e investigaciones.

Por tanto, para llevar a cabo la realización de este proyecto ha sido necesario realizar un estudio previo de los trabajos desarrollados en el campo de narices electrónicas y las distintas técnicas de modulación. Además, ha sido importante realizar un estudio de los sensores de tipo MOS, los cuales son usados en este proyecto, ya que son de vital importancia dentro de la plataforma y para la integración en el sistema de lazo cerrado que se pretende implementar.

La construcción de la plataforma parte de un modelo desarrollado previamente, sobre el que se realizan cambios en el hardware para conseguir que esta trabaje de forma versátil. La placa contiene los distintos circuitos para poder usar modulaciones basadas en técnicas de frecuencia y amplitud, así como también la implementación de la modulación en lazo cerrado a través de un controlador PID. La plataforma usará un solo sensor tipo MOS, el cual podrá conectarse en el correspondiente puerto de conexión según el tipo de modulación que se quiera realizar. En la construcción de la plataforma además se ha agregado un circuito de protección, para así evitar daños sobre la placa que se encarga del procesamiento de datos.

Por otra parte, la implementación de la técnica de modulación en lazo cerrado a través de un controlador PID supone un reto dentro del proyecto puesto que esta técnica aunque ya había sido desarrollada e implementada en un trabajo anterior, en este trabajo se lleva a cabo de forma distinta.

Tras finalizar este trabajo se consigue construir una primera plataforma de experimentación que ha servido para hacer las pruebas y luego se ha construida una segunda versión de la placa en la que se hicieron mejoras respecto a primera placa, cumpliendo cada una de ellas la misma funcionalidad.

Las plataformas de experimentación han sido diseñadas y equipadas para poder llevar a cabo experimentos con diferentes técnicas de modulación que previamente ya habían sido desarrolladas, además de implementar e integrar una modulación en lazo cerrado con un controlador PID.



## Palabras Clave

Nariz electrónica, Beagle Bone, sensores MOS, control PID, Python, TGS2600, modulación en lazo cerrado, discriminación de odorantes.

## Abstract

The aim of this project is the construction of a platform of electronic nose experimentation on closed-loop, which will be used for the acquisition and discrimination of different odorants under the study. Nowadays, development of the electronic noses is widely spread and in the last years, this field has been growing up to become in an active field of research. In order to conduct further experiments, the construction of this platform must be versatile and to allow the use of different modulation techniques which were implemented in previous works. Therefore, a previous study of the already existing work in the field of electronic noses and the different modulations techniques has been necessary. In addition, it have been important to carry out a study of the MOS sensors, which have been used in the project. They play an important role inside the platform and in the integration of the system closed-loop to deploy.

The construction of the platform starts with a previously protoboard design. Afterwards, changes in the hardware must be done to achieve a more versatile way of work. The platform contains the different circuits to use modulations based on the frequency and amplitude techniques. Also, it is included the implementation of the closed-loop modulation throughout a PID controller. The platform uses only one MOS sensor, which can be connected to the proper port depending on the chosen modulation. In the construction of the platform, a safe circuit has been added to avoid damage on the platform responsible for data processing.

Furthermore, the implementation of the closed-loop modulation technique throughout a PID controller is itself a challenge in the project due to the technique, even though it had been already developed and implemented in a previous project. In this design, it is done quite different. After finishing this project, a first platform of experimentation has been developed to perform the different tests. Afterwards, a second version of the platform has been deployed with improvements over the first version, but keeping the same functionality.

The platforms have been designed and implemented to perform different experiments with several modulation techniques previously developed. A closed-loop modulation with a PID controller has been implemented and integrated as well.



## Key words

Electronic nose, Beagle Bone, MOS sensors, PID control, Python, TGS2600, Closed loop modulation, odorant discrimination.

# Índice general

# Índice de figuras

# Índice de tablas

# Glosario

- **ADC**: Analog to Digital Convert
- **BBB**: BeagleBone Black
- **CI**: Circuito Integrado
- **GNB**: Grupo de Neurocomputación Biológica
- **GPIO**:General Purpose Input-Output
- **PID**: Proporcional Integral Derivativo
- **PCB**: Printed Circuit Board
- **PWM**: Pulse Width Modulation

# 1 Introducción

## 1.1. Motivación del proyecto

El sentido del olfato es un receptor químico en el cual las partículas aromáticas que son desprendidas de cuerpos volátiles, entran por el epitelio olfatorio el cual se encuentra ubicado en la nariz. La actividad eléctrica generada en el epitelio se transmite al cerebro donde la información es procesada por el sistema nervioso. El olor es una sensación debida a la percepción de un estímulo provocado por el sistema sensorial olfativo. El olor es generado por la mezcla de gases, polvo o vapores, en donde la percepción del olor está influenciada por la composición de la mezcla gaseosa, esta mezcla se denomina también odorante.

Las narices artificiales (EN's, Electronic Nose), se han desarrollado a lo largo de estos últimos años con el objetivo de ser capaces de comportarse como un receptor químico el cual pueda reproducir la capacidad de lectura e interpretación de los diferentes odorantes a nivel sensorial a los cuales son expuestos. El desarrollo de EN's a lo largo de los años ha contribuido a desafíos como son la localización de fuentes de odorante, el estudio de la distribución de gases en entornos, la detección de odorantes o discriminación de estos [1] [2], y la detección de ácido acético en muestras de vinos [3], etc.

Las herramientas básicas con las cuales se comenzó la experimentación de los estímulos olfativos se fundamentan en [2] [4] [5]. Una de estas herramientas ha sido el estudio de los sensores Fígaro TGS [6] los cuales son usados para el control de la calidad del aire, contaminantes atmosféricos, entre otros, lo que ha llevado a realizar diferentes estudios relacionados con él [7] [1] [8]. Esto ha sentado la base para llevar a cabo el estudio de aplicaciones de sensores odorantes integrados en robots [9], lo cual ha ayudado afrontar dos problemas como son la localización, y la división de la tarea haciendo uso de varios robots los cuales implementan lógica de localización para una única EN [10] [8].

Por otra parte, la discriminación de odorantes se ha llevado a cabo mediante el uso de técnicas de modulación de temperatura. En este tipo de modulación, tradicionalmente se fija una temperatura con la cual se controla el calentamiento del sensor y luego se lleva a cabo la lectura de la conductancia de este mismo. La modulación de temperatura es una forma de mejorar la capacidad de discriminación de los sensores, logrando que en la superficie del sensor se alteren las propiedades cinéticas, las cuales ayudan a conducir las distintas respuestas de los gases a los que es expuesto este mismo [1] [11] [12] [13] [14].

Los odorantes en general no se caracterizan por tener un único valor de conductividad, cada uno de ellos puede tomar diferentes valores cuando se aplican distintas temperaturas de calentamiento sobre el sensor, además de tener cuenta que estos valores de conductividad también dependen de la dinámica de cada odorante.

Por tanto, se tienen técnicas de modulación de temperatura activa en la cual la modulación va a depender de la respuesta del sensor en cada momento para llevar a cabo el proceso olfatorio. En este tipo de técnicas de modulación el objetivo es poder trabajar con los regímenes de operación dinámico, y de esta forma ajustar la temperatura para el odorante y la respuesta del sensor [1] [13] [5].

Tras el desarrollo de las técnicas de modulación para la discriminación de odorantes se desarrolla una plataforma de experimentación [5], usando una placa Beagle Bone Black (BBB) [15] con la cual se llevó a cabo el control de dicha plataforma.

Uno de los objetivos principales de este proyecto es implementar la técnica de modulación de temperatura activa e inversa en lazo cerrado desarrollada en [1] a través de un controlador PID, con lo cual se integrará el control de calentamiento de un sensor TGS2600 [6] sobre una placa de experimentación. En este caso, la técnica de modulación de temperatura se integrará sobre una placa diferente a la que se ha implemento en [1]. Para la consecución de este objetivo es necesario la construcciónón de una plataforma de experimentación, partiendo de un modelo en protoboard que se desarrolla en [5] la cual esta dotada de los elementos necesarios que requiere cada una de las técnicas de modulación. Por tanto, en una misma plataforma se podrán obtener los resultados de cada una de las modulaciones y así poder evaluar la respuesta obtenida del comportamiento de los odorantes bajo estudio. La construcción de esta placa hace que la implementación, ejecución y desarrollo de las técnicas de modulación sea mas sencilla gracias a la versatilidad de la placa.

El desarrollo, construcción y validación de la plataforma de experimentación y del circuito de protección así como la integración de otras técnicas de modulación y la implementación de la nueva técnica de modulación en lazo cerrado requiere conocimientos básicos de programación en Phyton, sistemas electrónicos digitales, conocimientos básicos de plataformas abiertas y versátiles. Por otra parte, es imprescindible conocer cómo funcionan los sistemas de control de lazo cerrado, específicamente el controlador PID [16], ya que es uno de los métodos usados en las técnicas de modulación de temperatura activa.

## 1.2. Objetivos

En este trabajo se quiere llevar a cabo el diseño y mejora hardware de una plataforma de experimentación, para la detección de odorantes usando técnicas de modulación de temperatura activa en lazo cerrado. Además del diseño de la plataforma también se integra un circuito de protección para esta misma.

En una primera etapa será necesario llevar a cabo un estudio previo sobre los trabajos realizados en el del Grupo de Neurocomputación Biológica (GNB) para poder tener los conocimientos necesarios acerca de las narices artificiales, así como también un estudio de la placa BBB y las plataformas creadas en trabajos anteriores.

Tras el estudio previo será necesario realizar la adecuación y mejora del hardware de la plataforma de experimentación, para así poder pasar a una etapa de diseño y la construcción de una placa en PCB (Printed Circuit Board). La plataforma soportará las diferentes técnicas de modulación a implementar. El diseño de la plataforma en PCB debe ser lo más versátil posible en caso de que haya cambios de componentes hardware. Después de la construcción de la plataforma se comprobará que esta funciona correctamente para las técnicas de modulación desarrolladas previamente [1] [13] [5].

Por otra parte, se llevará a cabo la implementación del software para la técnica de modulación activa e inversa en lazo cerrado a través de un controlador PID, y se pasará a validar los resultados obtenidos para esta nueva técnica de modulación.

## 1.3. Organización de la memoria

La organización de la memoria consta de lo siguientes capítulos:

- **Capítulo 1**: Introducción, este capítulo contiene la motivación y objetivos del proyecto, además de la organización de la memoria.
- **Capítulo 2**: Estado del arte, en este capítulo se aborda el marco teórico en el cual se encuentra el proyecto, se hace una breve introducción al sentido del olfato, así como algunas de sus características. Por otra parte, también se describe el estado de los sistemas actuales de detección de odorantes, además de estudiar los sistemas de control basados en el uso del controlador PID.
- **Capítulo 3**: Diseño y construcción de placa experimental, en este capítulo se explica el desarrollo y la construcción del prototipo de plataforma experimental, partiendo del diseño en protoboard que se realizó en [5]. Además, también se lleva a cabo mejoras del hardware que se han podido observar durante y después de la construcción del primer prototipo. Por tanto, al finalizar este capítulo se tendrá un diseño hardware de la plataforma que sea versátil y se adapte a las técnicas de modulación que se puedan agregar.
- **Capítulo 4**: Funcionamiento de la plataforma y pruebas, en este capítulo se busca comprobar el correcto funcionamiento de la plataforma creada en el capítulo anterior, para esto será necesario llevar a cabo pruebas con las técnicas de modulación desarrolladas en trabajos previos.
- **Capítulo 5**: Desarrollo y pruebas modulación en lazo cerrado, en este capítulo se llevará a cabo el desarrollo de una técnica de modulación en lazo cerrado a través de un controlador PID, partiendo del estudio realizado en [1]. En este caso, la técnica de modulación implementado debe ser capaz de seguir una función objetivo que se marque. Tras la implementación de la nueva modulación, se llevarán a cabo las pruebas necesarias usando la placa de experimentación que se ha construido en el capítulo 3 y se validaran los resultados.
- **Capítulo 6**: Conclusiones y trabajo futuro, en este capítulo se extraerán las conclusiones finales tras la finalización del proyecto, además de plantear nuevos objetivos con los que se pueden continuar trabajando sobre el proyecto realizado.

# 2

# Estado del arte

## 2.1. Introducción

Un olor es una sensación el cual es percibido por la interacción de odorantes (mezcla gaseosa de ciertos componentes químicos) con los receptores del epitelio olfativo, el cual se encuentra en la parte superior de la cavidad nasal. Los odorantes pueden ser compuestos volátiles e hidrofóbicos. El sentido del olfato es un sistema lo bastante sensible como para ser capaz de responder a concentración de compuestos químicos en muy bajas cantidades.

El sistema olfativo está organizado de igual forma que otro tipo de sistemas sensitivos como por ejemplo la visión, en este caso la entrada del sistema está siendo proporcionada por moléculas en forma gaseosa (odorantes). El sistema reconoce estos odorantes a través de proteínas receptoras las cuales están ubicadas en las membranas ciliares de las neuronas sensoriales olfativas en el epitelio olfativo como se muestra en la figura 2.1. El epitelio olfativo está compuesto por tres clases de células como son las neuronas olfativas primarias, células sustentadoras y células basales, figura 2.2. El epitelio olfativo se caracteriza por ser un tejido fino en la cavidad nasal, que en el ser humano puede ser difícil de distinguir. Los odorantes llegan a los receptores olfativos a través de las fosas nasales.

Desde el punto de vista de la ingeniería resulta complicado llevar a cabo el estudio del sistema olfativo, por lo que surge la necesidad de crear una herramienta artificial que permita medir de manera cuantitativa la presencia de un componente químico volátil. Esta herramienta son las denominadas narices electrónicas, las cuales son un dispositivo que tiene como función la captación, el análisis y la exploración de los diferentes odorantes que rodean nuestro entorno.

## 2.2. Narices electrónicas

Las narices electrónicas surgen por la necesidad de poder efectuar la detección y discriminación de odorantes. Este tipo de dispositivo está conformado por sensores químicos, los cuales son capaces de reconocer y comparar olores de forma individual. El objetivo principal de este dispositivo es el mismo que tiene el sistema olfativo humano, ser capaz de reconocer o relacionar el aroma, para así ser almacenado y poder usarlo en análisis posteriores.

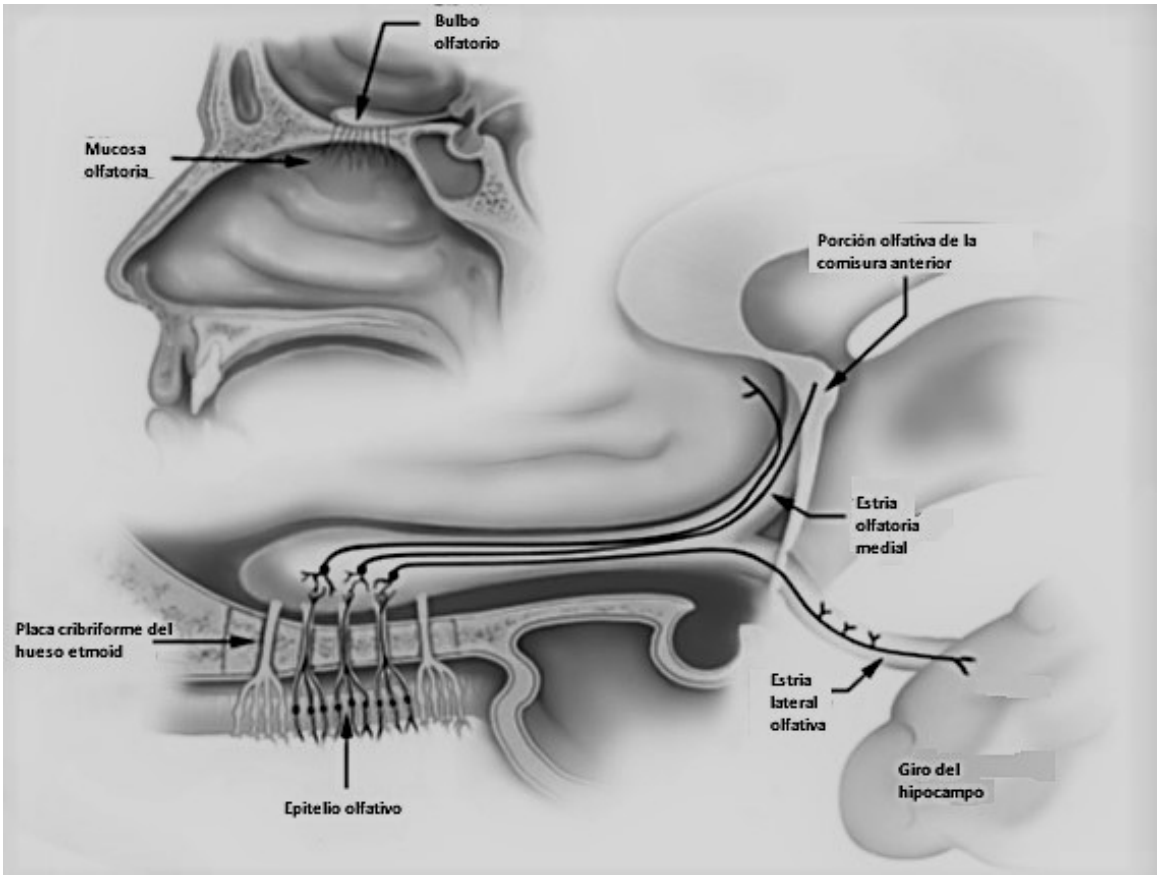



Figura 2.1: Anatomía sistema olfatorio, figura modificada de [17].

Los sistemas actuales de nariz electrónica constan de tres partes esenciales como son la toma de muestras, el conjunto de sensores y el sistema de tratamiento de los datos, como se muestra en la figura 2.3. La diferencia fundamental de las narices electrónicas que actualmente se pueden encontrar en el mercado, es el tipo de sensor empleado. En la actualidad lo sensores de gases son los que tienen mayor cavidad en el mercado. Los sensores de gases se constituyen de diferentes materiales (oxido metálico, polímeros conductores, cristales piezoeléctricos) los cuales les permiten modificar sus propiedades eléctricas en el momento que hay interacción con compuestos volátiles. Los principales tipos de sensores que se pueden encontrar son [17]:

- **Quimiorresistores**, este tipo de sensor se caracteriza por que sus propiedades de conducción eléctrica varían cuando detecta una sustancia química, por su baja selectividad lo que hace que la identificación de un único químico sea difícil. Por otra parte, este tipo de sensor tiene un coste módico, un tamaño reducido y es fácil de usar lo que hace que este tipo de sensor se haya usado en trabajos realizados dentro del GNB [2] [10] [5] [4] y por tanto también será usado en este proyecto.
- **Quimiotransistores**, este tipo de sensor está basado en transistores con un recubrimiento que hacer que sean sensibles a sustancias químicas. Comúnmente son usados en entornos líquidos y gaseosos
- **Quimiocapacitores**, este tipo de sensor está basado en un condensador el cual varia su capacidad ante la absorción de partículas en el dieléctrico
- **Quimiosensores amperimétricos**, este tipo de sensor mide la corriente que pasa a través de una celda electroquímica, aunque son muy sensibles tienen como desventaja su elevado

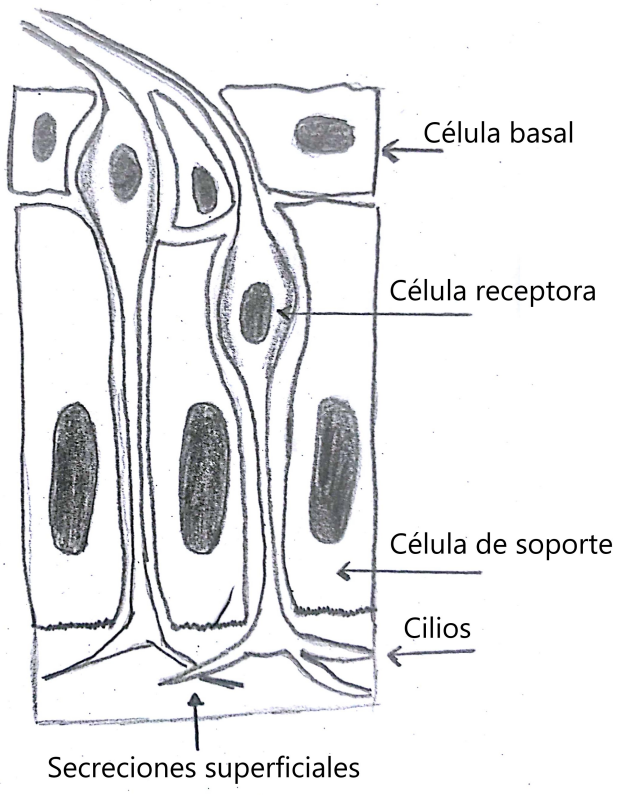



Figura 2.2: Epitelio olfatorio, figura modificada de [17].

coste.

### 2.2.1. Sensores Quimioresistivos

Los sensores quimioresistivos están basados en el cambio de conductividad MOS (Óxidos Metálicos Semiconductores) debido a la reacción química con las moléculas gaseosas, este tipo de sensor de gas es el más simple y el más usado para hacer medidas de odorantes [17]. Estos sensores se integran de forma sencilla dentro de un circuito eléctrico además de tener un bajo coste. El funcionamiento de estos sensores es el siguiente:

- Cuando el sensor se encuentra al aire libre de contaminantes, el oxígeno se acumula en la superficie del sensor. Los electrones libres de la superficie del sensor ante la presencia del oxígeno son atraídos creando una barrera de potencial (eVs para el aire). De esta forma, se crea una resistividad elevada debido a que el flujo de electrones se detiene, en la figura 2.4(a) se puede observar este proceso.

- Con la presencia del gas contaminante las partículas de oxígeno que se encuentra presente en la superficie del sensor son eliminadas. Por tanto, la densidad del oxígeno en la superficie se ve reducida y la barrera de potencial decrece lo que conlleva a que el flujo de electrones aumente y la resistividad del sensor disminuya. De esta forma, se consigue que al momento de hacer la medida de la resistencia del sensor se obtenga información tanto del gas que ha sido expuesto como de la concentración de partículas de este. Las reacciones químicas que se producen son modificadas tanto por la temperatura de calentamiento de la placa interna trasferida a la reacción como por los diferentes componentes reactivos que se pueden llegar

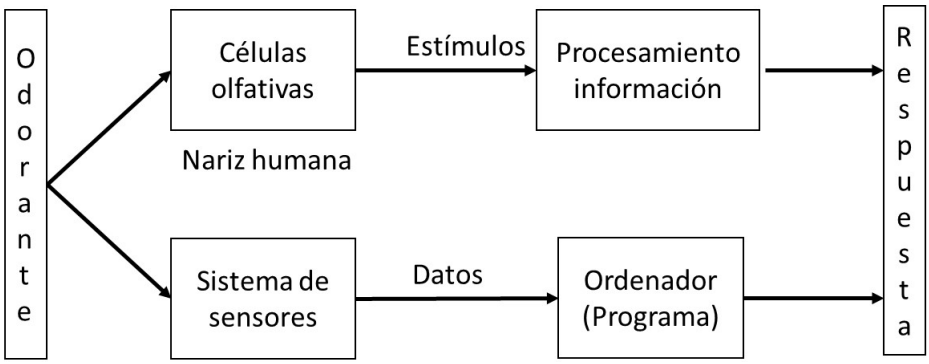
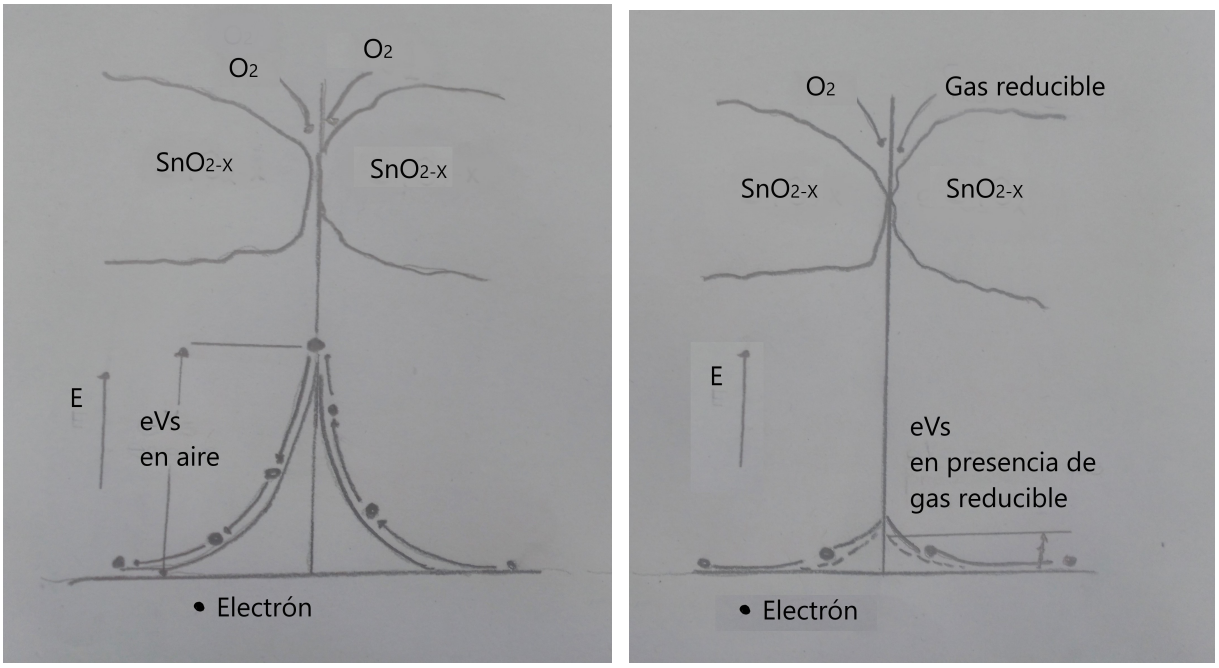



Figura 2.3: Proceso olfativo nariz humana vs nariz electrónica.

acumular en la superficie del mismo. En la figura 2.4(b) se observa el fenómeno que se ha descrito.




(a) (b)

Figura 2.4: (a) Esquema en el que se observa como el oxigeno se absorbe sobre la superficie de las partículas, donde se forma una barrera potencial en la frontera. (b) Esquema en el que se muestra la reducción de los átomos de oxígenos con gas reductor, haciendo que el flujo de los electrones sea más simple consiguiendo aumentar la conductividad. Imágenes adaptadas de [17].

En la tabla 2.1 se muestran algunos de los posibles sensores quimioresistivos que se pueden encontrar a nivel comercial.

Por otra parte, este tipo de sensor al poder integrarse fácilmente dentro de un circuito eléctrico tiene la ventaja que la señal de salida puede ser acondicionada para poder trabajar con los datos que devuelve el sensor. Como se ha comentado anteriormente, el cambio de conductividad en el sensor se vera reflejado tanto por la cantidad de odorante al que este expuesto como a la tensión de calentamiento que se aplique a este mismo. En este caso, la tensión de calentamiento es el parámetro de control que puede ser usado para poder llevar a cabo una modulación de temperatura activa. Por tanto, este sensor puede ser usado como una planta dentro de un sis-

| Fabricante | Aplicación | Modelo |
|---|---|---|
| FIGARO ENG | Gas combustible | TGS813, TGS816, TGS842, TGS821 |
| | Gas toxico | TGS203, TGS825, TGS826 |
| | Vapor de solvente | TGS822, TGS823 |
| | Gas halocarbono | TGS830, TGD831, TGS832 |
| | Control de calidad del aire | TGS800 |
| | Control de cocción | TGS880, TGS882, TGS883 |
| | Contaminantes Atmosféricos | GS2600 |
| NEW COSMOS ELEC.CO,LTD. | Gas combustible | CH-H, CH-M |

Tabla 2.1: Sensores quimioresistivos MOS disponibles comercialmente, tabla adaptada de [17].

tema de control, para así poder inspeccionar el cambio en la conductividad que puede llegar a experimentar el sistema al estar expuesto a las diferentes moléculas gaseosas, esto hace que este tipo de sensor se use dentro del desarrollo de este proyecto.

## 2.3. Sistemas de control

Un sistema de control se puede definir como aquel conjunto de componentes los cuales tienen la capacidad de regular su propia conducta o la de otro sistema, con el objetivo de lograr un comportamiento determinando. Los sistemas de control han tenido gran importancia dentro de distintos campos como los sistemas robóticos, vehículos espaciales, procesos tan sencillos como el control de temperatura en una habitación, flujo de un gas, entre otros.

### 2.3.1. Sistemas de control en lazo abierto

Un sistema en lazo abierto, se puede definir como aquel en el que la salida del sistema no ejerce ningún tipo de acción de control. Esto quiere decir que la salida del sistema no es comparada con la señal de referencia. La figura 2.5, muestra un ejemplo de como se comporta un sistema en lazo abierto. Por una parte en la entrada se tiene la señal de referencia que es conocida. Esta señal pasa a través del controlador el cual se encarga de ejercer la acción de control la cual se debe aplicar al sistema, por tanto, esta acción de control es la entrada del sistema. En el sistema pueden producirse perturbaciones externas que causan que el sistema no llegue a cumplir su función. El uso de un sistema en lazo abierto es recomendado cuando se conoce la relación entre la referencia y la salida del sistema.

### 2.3.2. Sistemas de control en lazo cerrado

Un sistema en lazo cerrado es aquel que se caracteriza por conservar la relación entre la entrada de referencia y la propia salida del sistema, por tanto la salida siempre es un dato de entrada. En la figura 2.6, se muestra un ejemplo de como funciona este tipo de sistema. En el controlador se observa como la entrada a este es el error, el cual relaciona la diferencia de la señal de referencia con la salida del sistema o realimentación. Así, el controlador ejercerá la acción de control a través del error para poder llegar a obtener a la salida el valor deseado. Dentro del sistema al igual que un sistema en lazo abierto también puede sufrir perturbaciones externas, al

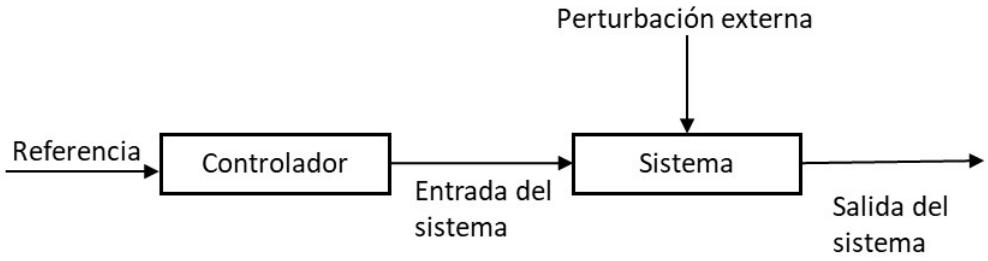



Figura 2.5: Diagrama de un sistema en lazo abierto.

usar un sistema en lazo cerrado se consigue que la respuesta de salida sea menos sensible a estas perturbaciones y a variaciones internadas debido a los parámetros del sistema.

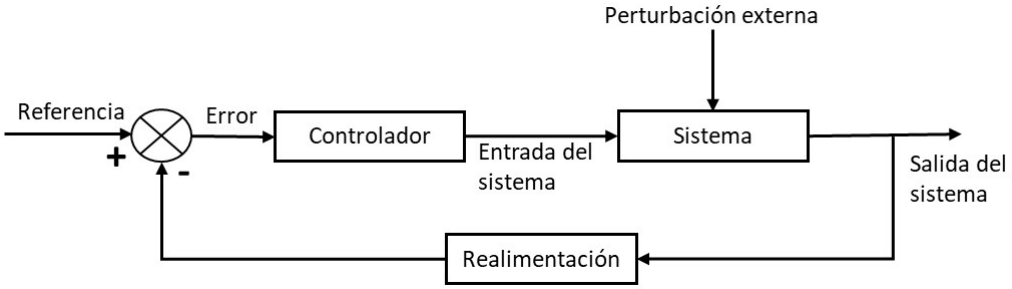



Figura 2.6: Diagrama de un sistema en lazo cerrado.

### 2.3.3. Controlador PID

El controlador PID es un sistema de control en lazo cerrado el cual integra tres acciones: proporcional (P), integral (I) y derivativa (D) [16]. Este tipo de control es el algoritmo más usado y el cual ha sido aceptado en el control industrial. Los controladores PID se caracterizan por su actuación robusta en un amplio rango de condiciones de operación y por su simplicidad, lo cual permite llevar a cabo un proceso de control relativamente simple y adaptativo en el tiempo.

Este es un sistema de control típico, figura 2.7, en el cual el proceso variable $y(t)$ es el parámetro del sistema que necesita ser controlado, por ejemplo la temperatura, presión o flujo de aire. Un sensor es usado para medir el proceso variable $y(t)$ y proveer una retroalimentación (que generalmente es el mismo proceso variable) para el control del sistema. La referencia $u(t)$ es el valor deseado para la variable del proceso. Por tanto, en cualquier momento el error $e(t)$ es decir:

$$e(t) = u(t) - y(t). \tag{2.1}$$

La diferencia entre la referencia y la variable del proceso es usado por el algoritmo del sistema de control (compensador o actuador) para determinar la salida del actuador que será la acción que se debe aplicar en la planta del sistema. Como se ha mencionado previamente el control PID aplica 3 acciones o coeficientes de control básicos, el proporcional ($K_p$), integral ($K_i$) y derivativo ($K_d$) lo cuales pueden variar para obtener la respuesta óptima.

- **Acción proporcional**: la componente proporcional es la acción mas sencilla y depende solo de la diferencia entre la referencia $u(t)$ y la variable del proceso $y(t)$. Esta diferencia

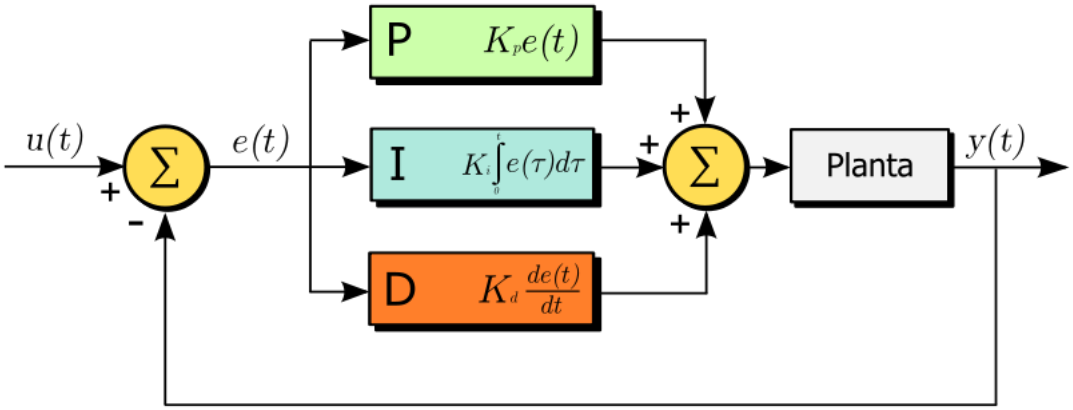



Figura 2.7: Diagrama controlador PID para un sistema de lazo cerrado. Imagen tomada de [18].

es denominada como el termino de error $e(t)$. La acción del control proporciona una salida en el controlador que es proporcional al error es decir, cuando el error es muy grande habrá una mayor actuación como se muestra en la ecuación 2.2.

$$P(t) = K_p \cdot e(t). \tag{2.2}$$

La ganancia proporcional ($K_p$) determina el ratio de la respuesta de salida. En general, incrementar la ganancia proporcional incrementara la velocidad de la respuesta del sistema de control. Sin embargo, si la ganancia proporcional es demasiado grande, el proceso comenzara a oscilar. Si se aumenta $K_p$, la oscilación llegara a ser más grande y el sistema volverá a ser inestable e incluso oscilara fuera de control.

- **Acción integral**: la componente integral suma el termino de error con el tiempo, es decir, no solo tiene en cuenta el error provocado por las posibles perturbaciones exteriores, si no que también tiene en cuenta el error acumulado en el sistema. La ecuación 2.3 refleja como se aplica la acción integral.

$$I(t) = K_I \cdot \int_0^t e(t) \cdot dt. \tag{2.3}$$

Esto da como resultado que, aunque el termino error sea pequeño la componente integral incremente lentamente. La respuesta integral continuamente incrementara en el tiempo al menos que el error sea cero, por lo que el efecto es conducir el error a un estado estable. El error del estado estacionario es la diferencia final entre la variable de proceso y la señal de referencia.

- **Acción derivativa**: la componente derivativa causa que la salida decrezca si la variable de proceso está incrementando rápidamente, la respuesta derivativa es proporcional a la tasa de cambio del proceso variable, es decir, se mide la velocidad de cambio del error, como se muestra en la ecuación 2.4.

$$D(t) = K_d \cdot \frac{de(t)}{dt}. \tag{2.4}$$

Incrementar el tiempo de derivación puede causar que el sistema de control reaccione de manera fuerte a cambios en el término de error y aumente la velocidad de la respuesta global del sistema de control. Sistemas más prácticos para el control del sistema usan tiempos derivativos muy pequeños ya que la respuesta derivativa es altamente sensible al ruido en la señal del proceso variable. Si la realimentación de la señal del sensor es ruidosa o si la tasa del bucle de control es muy baja, la respuesta derivativa puede causar que el sistema de control sea inestable.

La combinación de las tres acciones logra tener las ventajas de cada una de las acciones de forma individual. La ecuación de un controlador PID queda representada en 2.5

$$PID = K_p \cdot e(t) + K_I \cdot \int_0^t e(t) \cdot dt + K_d \cdot \frac{de(t)}{dt}. \tag{2.5}$$

En la tabla 2.2, se muestra de forma resumida como se puede jugar con los parámetros del controlador PID y como puede afectar a las características del sistema de control.

| | $K_p$ ***Aumenta*** | $K_i$ ***disminuye*** | $K_d$ ***aumenta*** |
|---|---|---|---|
| **Estabilidad** | Se reduce | Disminuye | Aumenta |
| **Velocidad** | Aumenta | Aumenta | Aumenta |
| **Error estacionario** | No eliminado | Eliminado | No eliminado |
| **Área de error** | Se reduce | Disminuye hasta cierto punto | Se reduce |
| **Perturbación de control** | Aumenta bruscamente | Aumenta gradualmente | Aumenta bruscamente |

Tabla 2.2: Reglas heurísticas de ajuste controlador PID, adaptado de [19].

### 2.3.4. Reglas de Ziegler-Nichols

En ocasiones, es posible encontrar un modelo matemático de la planta o sistema controlar, sobre el cual se quiere llevar a cabo el proceso de control, con lo cual se pueden aplicar diferentes técnicas para poder determinar cuáles son los parámetros del controlador que pueden cumplir las especificaciones del tiempo transitorio y estacionario de un sistema en lazo cerrado. Por otra parte, se puede dar el caso en el que no es fácil encontrar un modelo matemático que describa el comportamiento de la planta. Lo que conlleva a que encontrar los valores del controlador PID no sea posible a través de métodos analíticos. Para solucionar el problema de las plantas en las cuales no se puede encontrar un modelo matemático que describa su comportamiento, Ziegle y Nichols [20] sugirieron reglas para hallar los valores del controladores PID: $K_p$, $K_i$ y $K_d$. Para ello se basaron en la respuesta caracterizada del régimen transitorio en una planta. A continuación se describen brevemente los dos métodos de sintonía de Ziegles-Nichols [20].

#### 2.3.4.1. Método basado en la curva de reacción

En este método la idea es poder tratar la planta como un sistema en lazo abierto en el cual a la entrada del sistema se aplicará una entrada que será el escalón unitario.

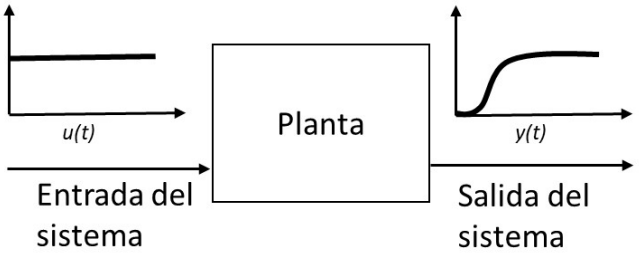



Figura 2.8: Respuesta del escalón unitario en un sistema en lazo abierto, figura adaptada de [16].

La entrada se obtiene de forma experimental o mediante simulación dinámica de la planta. A la salida de la planta la respuesta ante el escalón unitario debe tener una forma de S, si se obtiene una curva de salida como esta se podrá aplicar el método. La descripción de este método se observa en la figura 2.8, en donde se tiene a la entrada del sistema el escalón unitario y a la salida un curva en forma de S, la cual representa la salida esperada para poder aplicar el método.

La curva a la salida de la planta como se muestra en la figura 2.9, está caracterizada por dos parámetros, uno de ellos el tiempo de retardo $L$ y la constante de tiempo $T$. Estos dos parámetros se obtienen al pintar una recta tangente en el punto de inflexión sobre la curva de salida. De esta forma se determina la intersección de la recta tangente con el eje del tiempo y con la línea K.

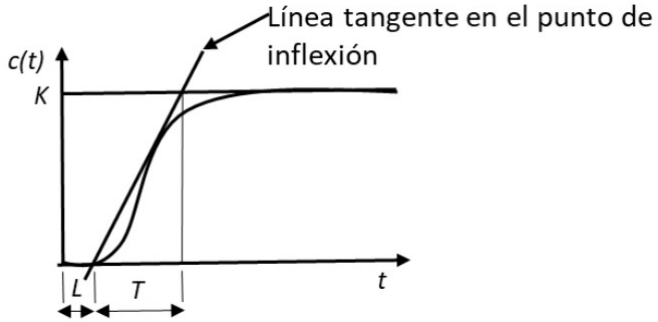



Figura 2.9: Curva de reacción a la salida del sistema ante la entrada del escalón unitario, figura adaptada de [16].

Ziegler y Nichols sugirieron establecer los valores para los parámetros $K_p$, $K_i$ y $K_d$ de acuerdo a los valores obtenidos de $L$ y $T$ como se muestra en la tabla 2.3

| Controlador | $K_p$ | $K_i$ | $K_d$ |
|---|---|---|---|
| P | $\frac{T}{L}$ | $\infty$ | 0 |
| PI | $0{,}9 \cdot \frac{T}{L}$ | $\frac{L}{0{,}3}$ | 0 |
| PID | $1{,}2 \cdot \frac{T}{L}$ | $2 \cdot L$ | $0{,}5 \cdot L$ |

Tabla 2.3: Parámetros de ajuste para el método de curva de reacción, tabla adaptada de [20].

#### 2.3.4.2. Método de oscilación

Este segundo método es válido solo para plantas estables en lazo abierto, el procedimiento a seguir es a través de un sistema en lazo cerrado como el que se muestra en la figura 2.10. Usando solo la acción proporcional, se comienza a incrementar el valor de $K_p$ desde un valor pequeño hasta un valor $K_{cr}$ (ganancia critica) en el cual se observe que la salida tenga oscilaciones. Por tanto, si a la salida no se presentan estas oscilaciones el método no se puede aplicar.

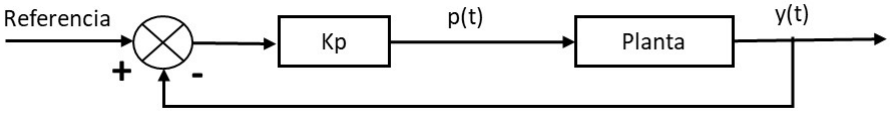



Figura 2.10: Sistema en lazo cerrado aplicando una acción proporcional, figura adaptada de [16]

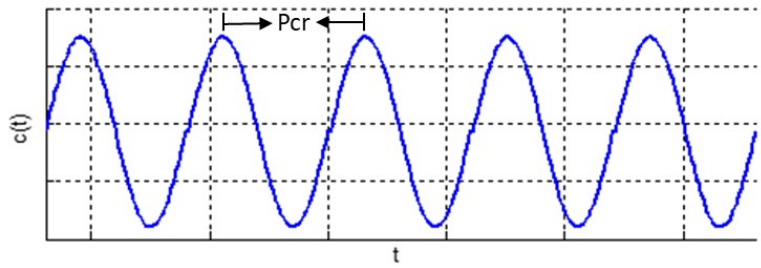



Figura 2.11: Respuesta con oscilación sostenida de periodo $P_{cr}$, figura adaptada de [16].

Una vez encontrado el valor de $K_{cr}$ también se obtendrá el periodo de la oscilación ($P_{cr}$) figura 2.11, estos dos valores son determinado de forma experimental. Ziegler y Nichols sugirieron los siguiente valores para los parámetros $K_p$, $K_i$ y $K_d$, como se muestra en la tabla 2.4.

| Controlador | $K_p$ | $K_i$ | $K_d$ |
|---|---|---|---|
| P | $0{,}5 \cdot K_{cr}$ | $\infty$ | 0 |
| PI | $0{,}45 \cdot K_{cr}$ | $Pcr \cdot \frac{1}{1{,}2}$ | 0 |
| PID | $0{,}6 \cdot K_{cr}$ | $0{,}5 \cdot P_{cr}$ | $0{,}125 \cdot P_{cr}$ |

Tabla 2.4: Parámetros de ajuste para el método de oscilación, tabla adaptada de [20].

# 3

# Diseño hardware y construcción plataforma de experimentación

## 3.1. Introducción

En este capítulo se desarrollará el proceso de diseño y construcción de la placa de experimentación partiendo del diseño en protoboard que se realizó en [5], además de explicar cada uno de los nuevos componentes hardware que se han seleccionado.

La plataforma de experimentación que se ha diseñado a lo largo de estos años dentro del GNB, es una plataforma la cual ha tenido distintos cambios tanto en su diseño como en sus componentes hardware. En este aspecto se debe tener en cuenta que la EN's está formada por un conjunto de módulos figura 3.1, los cuales interactúan entre sí para conseguir la percepción de los diferentes odorantes.

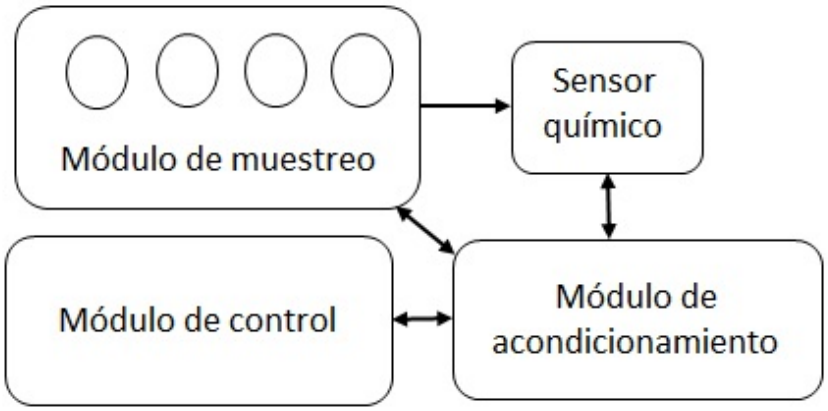



Figura 3.1: Módulos de la EN's

Al finalizar este capítulo se consigue la construcción de dos placas de experimentación. La primera que ha sido experimental para probar que todo funcionara correctamente y luego viendo lo errores de diseño se construyo una segunda placa que tiene como agregado el uso de una electroválvula más.

## 3.2. Módulo de control

Este módulo se encarga de trabajar con los datos que le llegan, por tanto, es usado para el procesado de los datos que van a ser adquiridos. Por otra parte, este módulo también se encarga de realizar el control de los elementos del sistema.

### 3.2.1. BeagleBone Black

La BBB está diseñada para la integración de software de alto nivel y electrónica de bajo nivel para la realización de cualquier proyecto. Una de las ventajas que tiene el uso de esta placa sobre placas tradicionales como Arduino, PIC entre otras, es el aprovechamiento del SO con el que cuenta la placa. Por tanto, el uso de la BBB es una buena opción cuando se quiere realizar un proyecto el cual puede llegar a ser demasiado complejo.

#### 3.2.1.1. Características generalas BBB

La placa BBB es un microcomputador el cual tiene hardware libre y que destaca principalmente por su baja potencia. Esta placa trabaja sobre un sistema embebido basado en Linux y por tanto esto hace que se pueda trabajar con código Open-source. En la tabla 3.1 se presentan las características más importantes que posee esta placa.

| | Carácterísticas |
|---|---|
| Procesador | Sitara AM33588BZCZ100, 1 GHz, 2000 MIPS |
| Motor gráfico | SGX530 3d, 20m Polygon/s |
| Memoria | SDRAM 512 MB. DDR3L 800MHz |
| Onboard flash | 4GB, 8bits Embedeed MMC |
| PMIC | TPS65217C PMIC regulator and one additional LDO |
| Alimentación | miniUSB or DC Jack, 5VDC External Via Expansion Header |
| Indicadores | 1-Power, 2-Ethernet, 4-User Controllable LEDs |
| HS USB 2,0 Client Port | Access to USB0, Client mode via miniUSB |
| HS USB 2,0 Host Port | Access to USB1, Type A Socket, 500mA LS/FS/HS |
| Puerto serie | UART0 access via 6 pin 3.3V TTL Header. Header is populated |
| Ethernet | 10/100, RJ45 |
| SD(MMC Conector) | microSD, 3.3V |
| Entradas de usuario | Reset, Boot y Power Bottom |
| Salida de video | 16b HDMI, 1280x1024 (MAX)<br>1024x768, 1280x720, 1440x900, 1920x1080@24Hz w/EDID Support |
| Audio | Via HDMI Interface, Stereo |
| Conectores de expansión | Power 5V, 3.3V, VDD_ADC(1.8V)<br>3.3V I/O on all signals<br>McASP0, SPI1, I2C, GPIO(69 max), LCD, GPMC, MMC1, MMC2,<br>7 AIN(1.8V MAX), 4 Timers, 4 Serial ports, CAN0,EHRPWM(0.2),<br>XDMA Interrupt, Power Button,<br>Expansion Board ID (Up to 4 can be stacked) |
| Peso | 1.4 oz (39.68grams) |

Tabla 3.1: Características BBB.

Esta placa tiene diferentes elementos dentro de su diseño y que se explicará a continuación, cada uno de estos elementos se podrá ver en la figura 3.2:

- Botón reset.

- Botón de encendido y apagado.
- Puerto USB, el cual proporciona acceso a través del USB HUB, y al conectarse al ordenador será un dispositivo estándar.
- Puerto de alimentación, la placa se puede alimentar con una fuente externa de 5 VDC o a través del puerto USB. Solo se puede alimentar con una de las dos opciones, no con las dos a la vez.
- LEDS indicadores, la placa posee 5 LEDs los cuales indican si la placa está siendo alimentada, además de poder ser usados por el usuario.

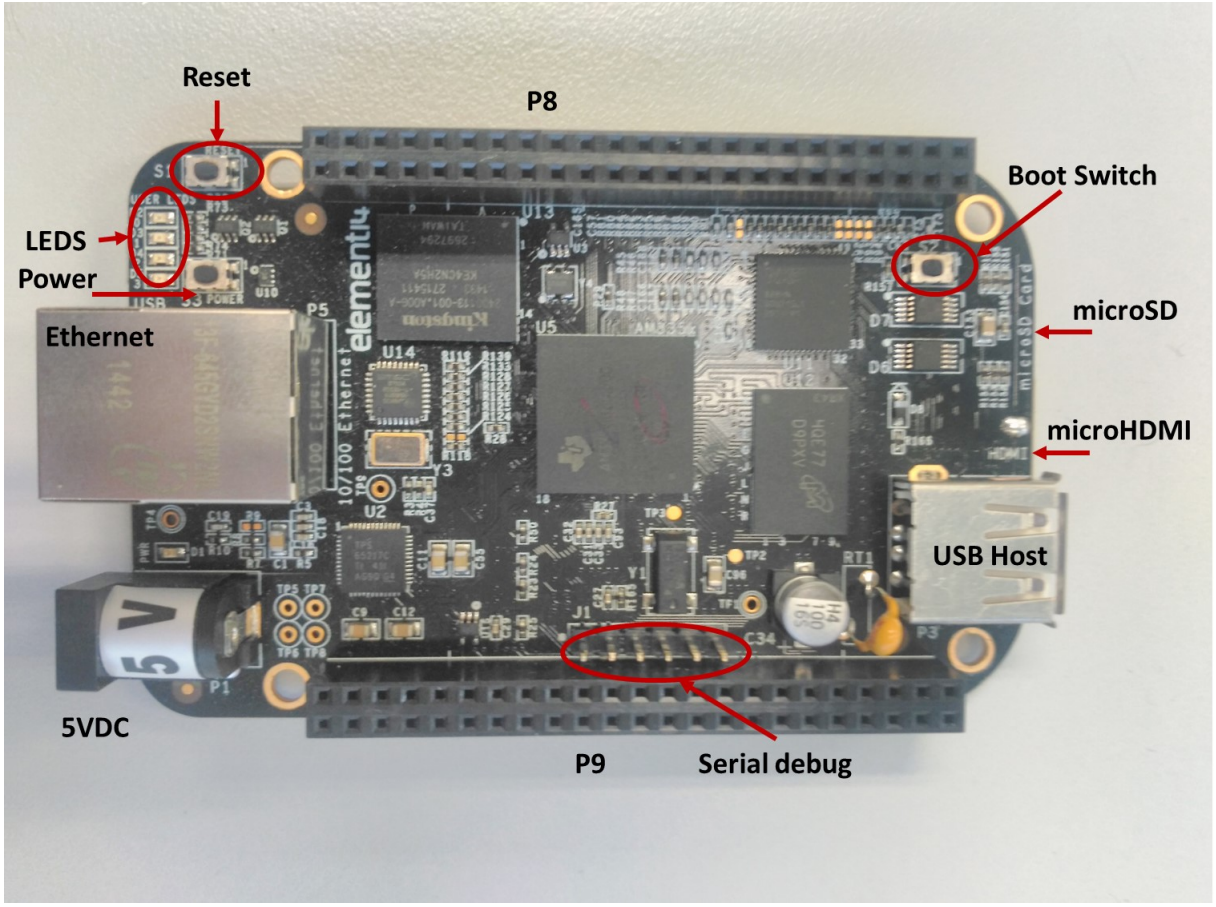



Figura 3.2: Conectores, interruptores y puertos de expansión BBB

La BBB tienes dos pines de expansión (P8 y P9) con 96 pines disponibles, los cuales tienen como ventaja que permiten conectar a otros sistemas periféricos de tal forma que se pueda recibir y enviar información. En este proyecto los pines adicionales son usados para obtener las siguientes funcionalidades:

- ADC, dentro la placa se pueden encontrar siete pines específicos (AINX) para realizar la conversión de señales analógicas a digitales. Tiene una tensión de conversión de 1,8 V como entrada máxima con una resolución de 1 mV. La salida digital tiene una resolución de 12 bits.
- GPIO, en la placa hay 65 pines disponibles (GPIOX) los cuales generan un '1' lógico con una tensión de 3,3 V.

- PWM, cuenta con seis salidas PWM con alta resolución y otras dos con modo eCap (enhanced Capture).

Es importante tener en cuenta los límites de trabajo que pueden tener algunos de los pines de la placa, ya que si se superan estos rangos posiblemente la BBB se pueda estropear. En la tabla 3.2 muestra los rangos para los pines analógicos y digitales de la placa

| I/O | Nombre | No. pines | Vmin-Vmax | Imax | TSample |
|---|---|---|---|---|---|
| Analog(I) | AINX | 7 | 0 - 1,8 V | 2 uA | 125 ns |
| Digital (I) | GPIOX | 66 | 0 o 3,3 V | 4 - 6 mA | 140 ns |
| Digital (O) | GPIOX | 66 | 0 o 3,3 V | 4 - 6 mA | 95 -105 ns |

Tabla 3.2: Limitaciones Placa BBB.

Otros periféricos que se pueden conectar a la placa a través de los pines de expansión son:

- Placa LCD de 24 o 16 bits.
- Dos puertos SPI (Serial Peripheral Interface).
- Puerto serie, la placa dispone de cuatro puertos serie en los conectores de expansión (UART1, UART2, UART3, UART4).
- Dos interfaces de bus CAN.
- Cuatro temporizadores de salida.

#### 3.2.1.2. Recomendaciones de uso BBB para su cuidado

La placa BBB es un dispositivo complejo y delicado el cual puede estropearse si no se usa de forma correcta. Es importante tener cuidado en el momento de conectar algún circuito externo a la placa ya que si se sobrepasan los límites de tensión o corriente que recomienda el fabricante el microprocesador se puede ver afectado y por será necesario comprar una nueva placa, ya que el micro no se puede reemplazar. Algunas recomendaciones de uso son las siguientes:

- No apagar la BBB desconectándola directamente de la fuente de alimentación. La placa se debe apagar correctamente usando el software de apagado o en su defecto presionar el botón de apagado durante algunos segundos. En el anexo B.4 se explica como se debe apagar la placa correctamente.
- No poner la placa sobre superficies metálicas o en encimeras en las cuales puede haber cables, resistencias, etc. Se recomienda usar un contenedor para poner la placa como el que se muestra en la figura 3.3.
- No conectar circuitos que puedan estar encendidos a los pines de expansión P8 y P9 mientras la BBB está apagada. Es importante asegurarse que todos los circuitos de interconexión hacia la placa suministren como máximo 3.3 VDC.
- Comprobar cuidadosamente el número de pin que se quiere usar, cada bloque de pin de expansión cuenta con 46 pines, y puede ser fácil confundirse y conectar en el pin que no es el correcto.

La información que se brinda en este proyecto acerca de la placa es solo una parte de todo lo que puede saber de esta misma, en [15] [21] [22] se puede encontrar más información detallada del uso de la placa.

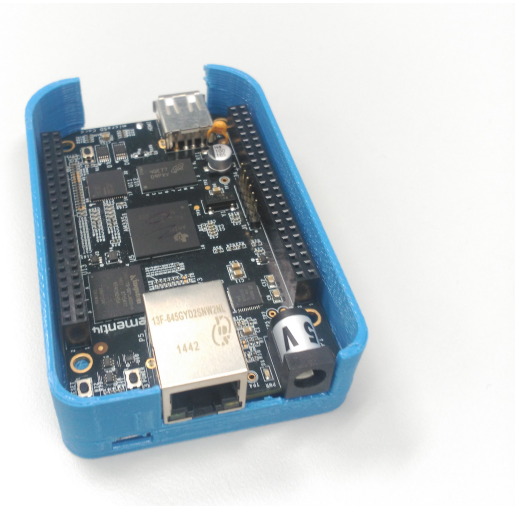


Figura 3.3: Caja de protección para la BBB.

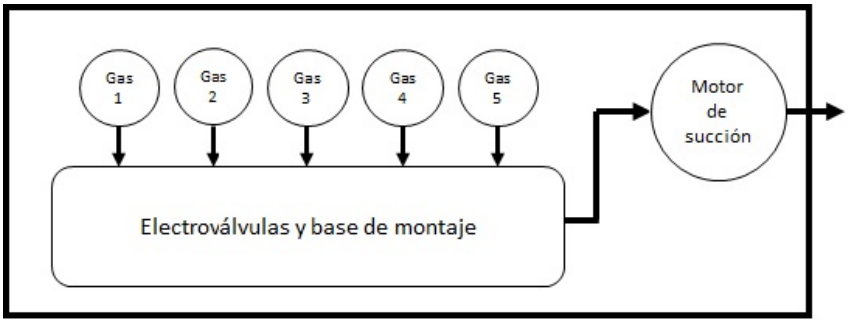
## 3.3. Módulo de muestreo

Este módulo contiene las muestras de los odorantes con las cuales se va trabajar, así como los elementos que ayudan a realizar el proceso de succión de cada uno de los odorantes, en la figura 3.4, se muestra los elementos que componen este módulo así como su interación.

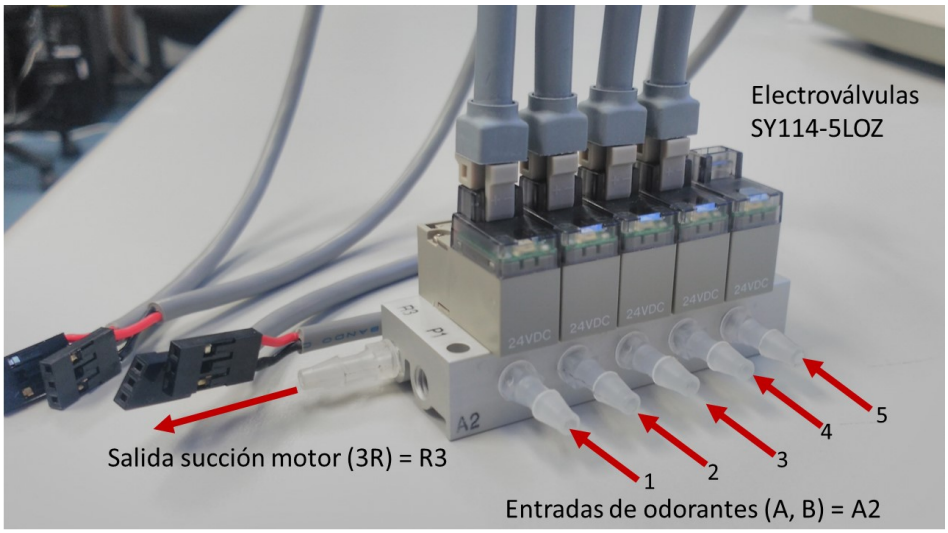
### 3.3.1. Electroválvulas

En las dos plataformas que se han hecho se usan 4 y 5 electroválvulas respectivamente. En trabajos anteriores para el desarrollo de la plataforma [5] [2] se han usado el mismo modelo electroválvula. Para este trabajo se han usado válvulas del mismo fabricante, pero con distintas características sobre todo por la tensión de alimentación que usan estas. Las electroválvulas usadas son SY114-5LOZ, figura 3.5(a), del fabricante SMC [23]. Estas electroválvulas de solenoide son de 3 vías, 2 posiciones y normalmente cerrada (NC). Este componente se debe montar sobre una base, la cual se detalla en la sección 3.3.2. Estas electroválvulas ofrecen un bajo consumo además de tener una vida útil de 100 millones de ciclos. A continuación, se describen algunas de las características principales de este dispositivo:

- Rosca del Puerto de Conexión: M3 métrico x 0,5.
- Tipo de Actuación: Solenoide/Resorte.
- Tensión de Solenoide 24 V dc.
- Consumo de Alimentación del Solenoide 0.55 W.
- Mínima Presión de Funcionamiento 0 MPa.
- Máxima Presión de Funcionamiento 0,7 MPa.
- Mínima Temperatura de Funcionamiento -10 ºC.
- Máxima Temperatura de Funcionamiento +50 ºC.
- Número de Puertos 3 vías/2 posiciones.




(a)




(b)

Figura 3.4: (a) Esquema de modulo de muestreo. (b) Sistema de electroválvulas actual de la EN.

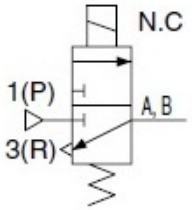
El funcionamiento de este dispositivo teniendo en cuenta el diagrama de la figura 3.5(b), es el siguiente:

- Cuando se aplica energía a la electroválvula, por tanto hay paso de corriente, esta conmuta y se permite el paso del fluido o el gas entre 1(P) y(A, B).
- Cuando no se aplica energía a la electroválvula, por tanto no hay paso de corriente, el dispositivo se encuentra en posición cerrada, y el paso del fluido o gas se realiza entre las puertas (A, B) y 3(R).

### 3.3.2. Base de montaje electroválvulas

La base de montaje de las electroválvulas es uno de los cambios que se ha realizado en el diseño de este prototipo y son un complemento necesario para poder usar las electroválvulas. Previamente las bases usadas para las electroválvulas eran bases individuales, su funcionamiento se explica en [5, ver 4.5.1]. Este tipo de base no era el adecuado para el funcionamiento que se requería, puesto que con estas bases se debió crear un array en el cual estas se pudiesen conectar de forma consecutiva. De esta forma se lograba que las electroválvulas pudieran conmutar entre el estado de energía o sin energía, de tal forma que se permitía el paso del fluido o gas conectado a cada electroválvula.

(a)

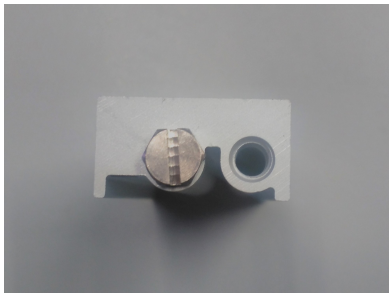



(b)

Figura 3.5: Electroválvula usada en el proyecto: (a) Electroválvula SY114-5LOZ. (b) Esquema electroválvula.

Tras investigar un poco más a fondo sobre los productos ofrecidos por el fabricante SMC se encontró un base de montaje la cual se presenta en la figura 3.6. Para saber si este tipo de base podría cumplir el funcionamiento deseado fue necesario ponerse en contacto con el fabricante para conseguir más información del producto y su funcionamiento, ya que el distribuidor no ofrecía más información acerca de este. Después de comentarle al fabricante lo que se estaba buscando y el funcionamiento que se quería obtener, el fabricante sugirió que el producto se ajustaba a las necesidades del proyecto, además, de explicar cuál era el uso correcto de esta nueva base.


(a)



(b)


(c)

Figura 3.6: Base de montaje electroválvula SS3Y1-S41-05-M5-QSMC.

Esta base usa la misma terminología que las electroválvulas, como se muestra en la figura 3.5(b) con respecto al nombre de las aperturas $A2(A, B)$, $P1(1P)$ y $R3(3R)$. En la figura 3.6(a) se observa varios orificios sobre la base, en esta parte de la base es donde se debe conectar las electroválvulas. En la figura 3.6(b), se observa dos orificios uno de ellos está abierto y el otro cerrado con un tornillo. Estos dos orificios ($P1$ y $R3$) atraviesan toda la base de un extremo a otro, y siempre uno de ellos debe estar cerrado, ya que el orificio cerrado será el común por donde podrá circular el gas que se vaya a succionar. En la figura 3.6(c), se observan 5 orificios($A2$) sobre la base , estos serán los puntos de interconexión con los odorantes, por tanto, en cada orificio se tendrá conectado un odorante diferente. En la figura 3.7 se observa el montaje de la base con las electroválvulas.

Para conseguir el funcionamiento que se busca en el sistema, el fabricante recomienda que las electroválvulas se usen sin aplicar energía por tanto esto significa que el dispositivo se encuentra en posición cerrada, y el paso del fluido o gas se realiza entre las puertas (A, B) y 3(R) de las electroválvulas que se corresponde con $A2$ y $R3$ en la figura 3.4(b). Por tanto, en la base de montaje la apertura $R3$ debe ir cerrada con el tornillo en uno de sus extremos ver figura 3.6(b), y

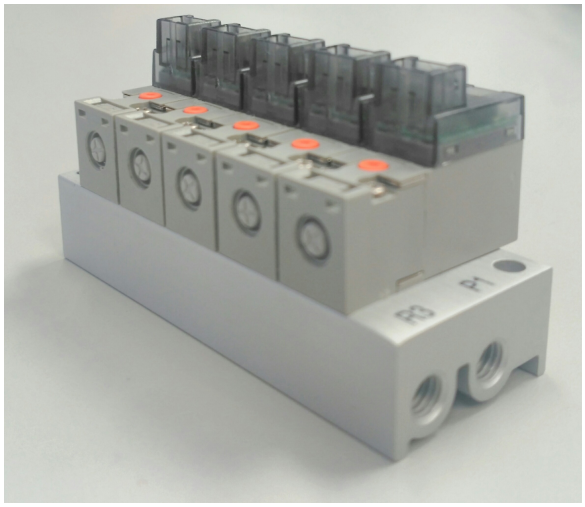



Figura 3.7: Montaje del conjunto de electroválvulas sobre la base.

en el otro extremo es por donde saldrá el odorante que se succione el cual ira hacia el sensor. Por ejemplo, si se quisiera dejar el paso del gas que esta conectado en la entrada 1 de la figura 3.4(b), se debe aplicar corriente en las entradas 2,3,4 y 5, de esta forma el flujo del gas de la entrada 1 ira hasta $R3$ es donde esta la salida de succión del motor.

### 3.3.3. Motor de succión

Este es uno de los elementos más importantes dentro del módulo de muestreo, ya que es el encargado de realizar el proceso de succión del odorante. Como se menciona en [5], la elección de este dispositivo se ha hecho de forma aleatoria y para la construcción de esta plataforma se ha elegido un modelo similar al que usado en [5]. En este caso el motor seleccionado ha sido el D250-BL [24] para la primera placa de experimentación y el 220-BL [24] para la segunda placa de experimentación, las diferencias de estos motores están en las características de la presión de funcionamiento y la tase de flujo. En la figura 3.8 se puede observar el motor, el cual es una bomba de gas y liquido de micro diafragma construido por RS. Este tipo de motor es usado para aplicaciones como flujo de presión, sistemas analíticos, detección y control de gases, sistemas de refrigeración, entre otros.

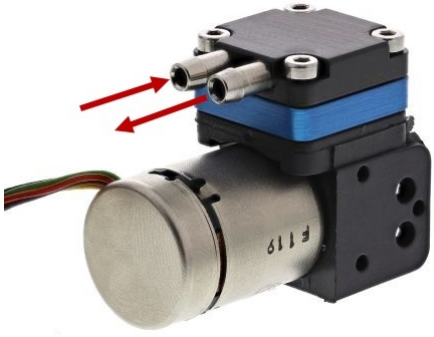


Figura 3.8: Bomba de desplazamiento positivo D250-BL y 220-BL.

Este dispositivo se caracteriza por ser eficiente y ligero, su construcción sólida y la amplia tolerancia a temperaturas de trabajo, permite que pueda actuar de forma confiable y trabajar en ambientes hostiles. En la tabla 3.3 y 3.4, se muestran las características de succión del motor D250BL y 220-BL respectivamente.

| | Input Voltage (V) | Power Usage (w) | Max Pressure (psi) | Max Free Flow (mlpm) |
|---|---|---|---|---|
| **Aire** | 1.5 - 5 | 0.1 - 0.6 | 12 | 620 |
| **Agua** | 1.5 - 5 | 0.6 - 1.8 | 9 | 100 |

Tabla 3.3: Características motor D250BL.

| | Input Voltage (V) | Power Usage (w) | Max Pressure (psi) | Max Free Flow (mlpm) |
|---|---|---|---|---|
| **Aire** | 1.5 - 5 | 0.1 - 0.6 | 11 | 450 |
| **Agua** | 3.0 - 4.5 | 0.6 - 1.8 | 8 | 90 |

Tabla 3.4: Características motor 220-BL.

El dispositivo funciona con tensiones entre 1,5 y 5 V y una corriente de 0.5 A. Además, puede ser conectado a baterías, su conexión es bastante sencilla, en la tabla 3.5 se muestra la respectiva conexión de este, para así evitar que el motor se pueda estropear. La velocidad de bombeo se puede ajustar variando la tensión de entrada que sugiere el fabricante.

| Color del cable | Funcion | Descripción | Conexión |
|---|---|---|---|
| Negro | GND | Conexíon negativa | Obligatorio |
| Gris | Motro Power | Control de velocidad del motor (1,5-5 VDC) | Obligatorio |
| Rojo | Vcc | 1,5-5 VDC | Obligatorio |
| Verde | Rotación Motor (CW/CCW) | Conectado a Vcc | Obligatorio |
| Amarillo | RPM | GND /Vcc 4 pulsos | Opcional |

Tabla 3.5: Conexión del motor para su funcionamiento.

## 3.4. Sensor químico TGS2600

Este elemento se encarga de realizar la adquisición de los odorantes bajo estudio. Este sensor varía su respuesta en función de la cantidad de odorante detectado en el momento de realizar el proceso de succión.

El sensor está compuesto por una capa semiconductora de óxido metálico, un chip de detección y un calentador el cual está integrado. Ante la presencia de gases, la conductividad del sensor incrementa dependiendo de la concentración de gas que haya en el aire.

Este tipo de sensor reacciona de forma diferente ante la presencia de gases, esto hace que las respuestas de los sensores sean diferentes en ocasiones ante la presencia de un mismo gas. Esto es debido al proceso de precalentamiento que se debe llevar a cabo antes de que estos sean usados, para así poder alcanzar un valor de resistencia estable y eliminar impurezas presentes en el interior del sensor. También cabe destacar que este tipo de sensor tiene memoria.

El sensor TGS2600 figura 3.9, es un sensor químico usado para la detección de contaminantes en el aire, este se caracteriza por:

- Su bajo consumo.


Figura 3.9: Sensor TGS2600, figura tomada de [6].

- Alta sensibilidad a los gases contaminantes presentes en el aire como hidrogeno y monóxido de carbono.
- Tiene una vida útil larga.
- Bajo coste, ya que es un componente que se fabrica en altas cantidades hace que su coste no sea excesivo.
- Fácil de usar, gracias a un simple circuito eléctrico se puede convertir el cambio de la conductividad en una señal de salida que se corresponda con la concentración del gas.
- Tamaño pequeño.

Un punto importante de este sensor es la sensibilidad que posee ya que esta viene dada por la relación que existe entre los cambios de la concentración de gas y la variación de la resistencia del sensor. En la figura 3.10 se observa al curva característica de sensibilidad del sensor TGS260 en la cual sobre el eje de abscisas se representa la concentración de diferentes gases y sobre el eje de ordenadas su respectiva relación del valor medido de la resistencia del sensor ($R_S$) para los gases mostrados en diversas concentraciones comparándolo con la medida de resistencia del sensor al aire libre ($R_O$).

Por otra parte, también se debe tener presente los efectos que pueden causar la temperatura y la humedad sobre el sensor puesto que pueden afectar de forma directa a la sensibilidad de este mismo. Esto es debido a que la detección de gases en el sensor se lleva a cabo a través de un proceso de adsorción y desorción en la superficie del sensor lo que puede provocar cambios en la magnitud de las reacciones en la superficie del sensor. En la figura 3.11 se puede observar como el cambio de temperatura puede provocar un cambio en la resistencia del sensor. Para el uso del sensor TGS2600 es recomendable usar un sensor de temperatura y humedad para poder controlar estos dos parámetros.

### 3.4.1. Circuito eléctrico de medida

Como se ha comentado previamente este sensor se puede usar gracias un sencillo circuito eléctrico para así poder transformar la conductividad en una señal de salida que se corresponda con la concentración de gas, este circuito se observa en la figura 3.12.

El circuito requiere de dos entradas de tensión, una de ellas es la tensión del calefactor ($V_H$) y la otra la tensión del circuito ($V_C$). La tensión del calefactor ($V_H$), se suministra al calefactor integrado para así poder mantener el elemento de sensado a una temperatura óptima para llevar a cabo el proceso de medida. La tensión del circuito ($V_C$) es aplicada para permitir la medida de la tensión de salida ($V_{OUT}$) a través de una resistencia de carga ($R_L$) la cual se conecta en serie con el sensor. Por tanto, el valor de tensión de salida del circuito se obtiene a través de un divisor de tensión como se muestra en la ecuación 3.1.

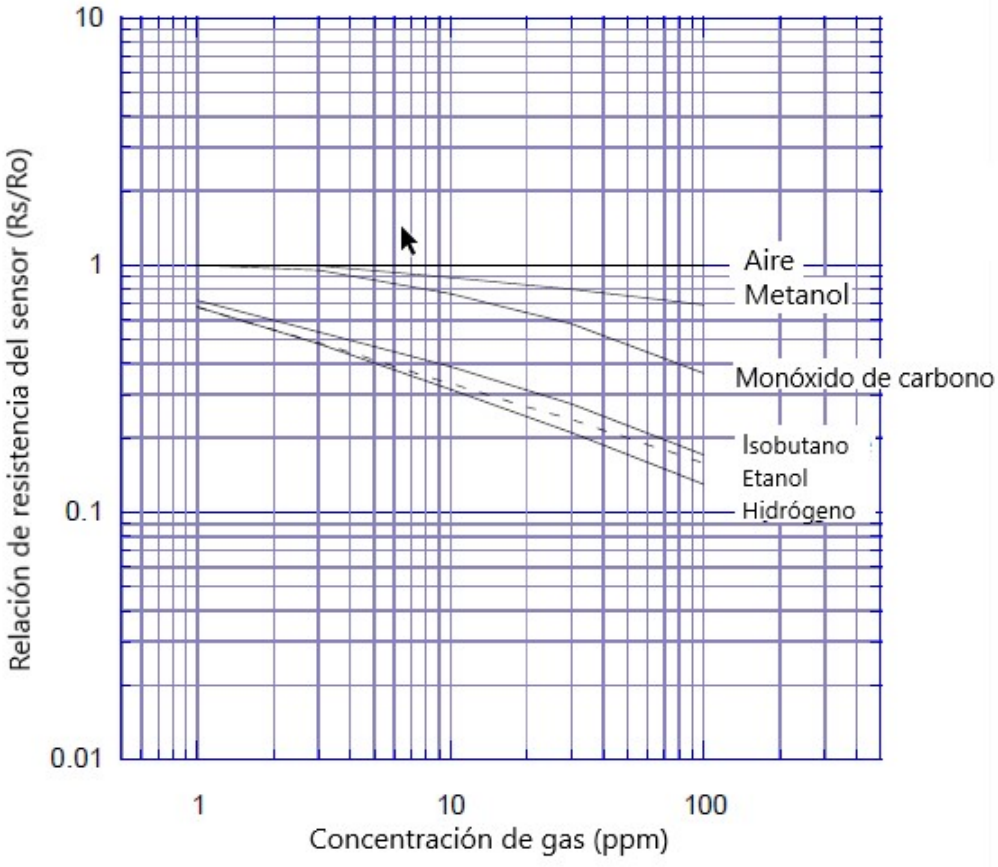



Figura 3.10: Curva de sensibilidad del sensor TGS2600. Imagen adaptada de [6].

$$V_{OUT}(R_L) = V_C \cdot \frac{R_L}{R_L \cdot R_S}. \tag{3.1}$$

En este caso, el valor de la resistencia de carga ($R_L$) es el parámetro que se debe seleccionar y que determinara los valores de tensión máximo y mínimo a la salida del sensor, el valor de resistencia de carga usado para este proyecto es de 470Ω, el cálculo de este valor se puede ver en [5]. En este caso, el valor de esta resistencia de carga se ha seleccionado teniendo en cuenta la tensión de entrada máxima de los pines analógicos de la placa BBB. En la tabla 3.6, se pueden encontrar especificaciones estándar y eléctricas del sensor TGS2600.

| **Especificaciones TGS2600** | | | |
|---|---|---|---|
| Condiciones estándar del circuito | Tensión de calefactor($V_H$) | $5,0 \pm 0,2$V DC/AC | |
| | Tensión del circuito ($V_C$) | $5,0 \pm 0,2$ V DC | $P_s \geq 15$ mW |
| | Resistencia de carga ($R_L$) | Variable | $0,45K\Omega$ |
| Características eléctricas bajo condiciones de test | Resistencia del calefactor ($R_H$) | 83Ω aprox | |
| | Resistencia del sensor($R_S$) | $10K\Omega \sim 90K\Omega$ en aire | |

Tabla 3.6: Especificación TGS2600, adaptado de [6].

### 3.4.2. Cápsula para el sensor TGS2600

El sensor TGS260 [6] como se puede ver en la figura 3.9, se encuentra dentro de un encapsulado y por tanto el sensor siempre estará expuesto a cualquier odorante o componente químico

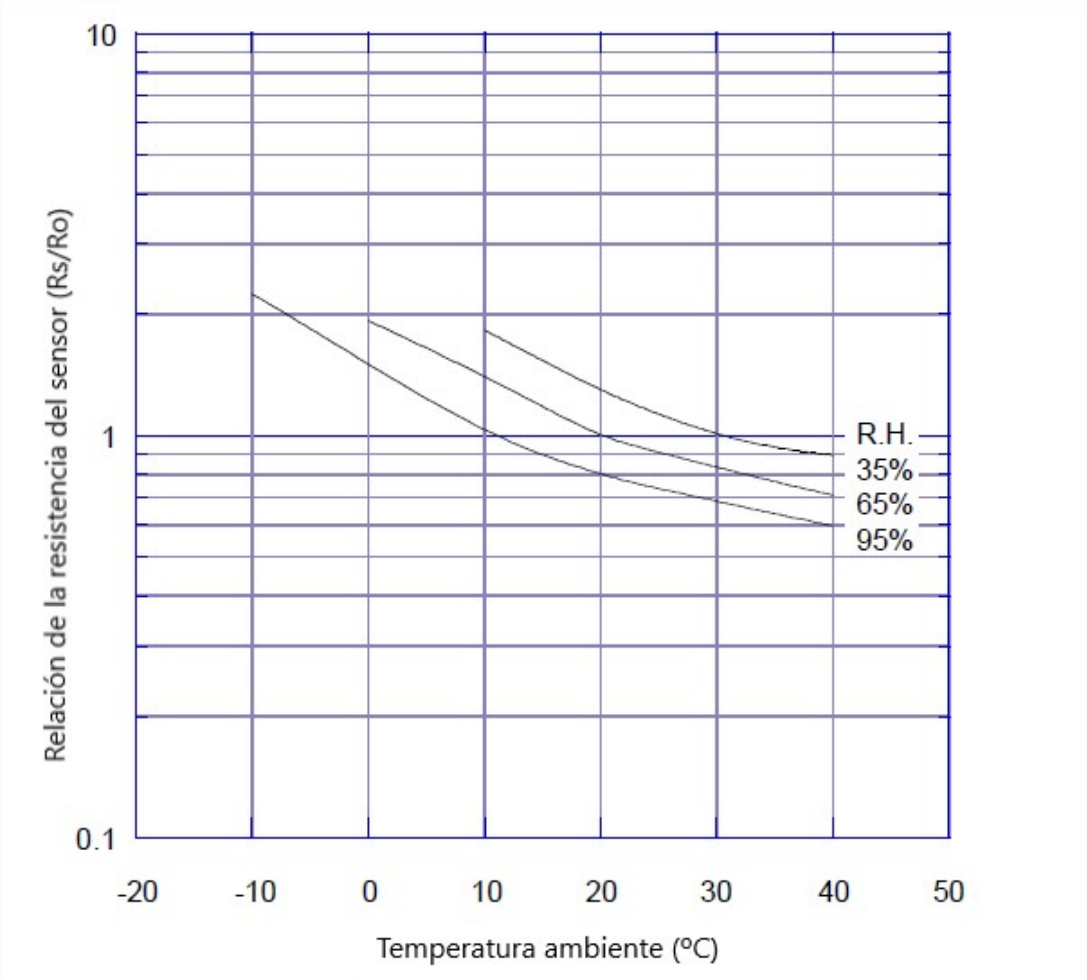



Figura 3.11: Dependencia de la resistencia del sensor frente a la temperatura y la humedad. Imagen adaptada de [6].

presente en el medio en el que se encuentre el sensor. Por tanto, ha sido necesario construir una cápsula en la cual se ubique el sensor para que así este solo expuesto a ciertos odorantes.

La cápsula para el sensor es una caja de plástico inerte y hermética que se ha diseñado para poder ubicar el sensor en una posición en la que este expuesto a los odorantes cuando se produce la succión. Este cápsula se puede observar en la figura 3.13, se ha construido usando un tubo de plástico específicamente el tubo de un rotulador. La cápsula esta construida de tal forma que el odorante que entra por la conexión a la salida $R3$ de la base de montaje de las electroválvulas, ver sección 3.3.2, impacte de frente con el sensor. La cápsula tiene una salida la cual esta conectada directamente con el motor que es el punto por donde se produce la succión. Con la construcción de esta cápsula se consigue que el odorante se concentre dentro de la cápsula cuando se lleva a cabo el proceso de succión además de conseguir que estén expuestos de forma directa con el sensor.

## 3.5. Sensor de temperatura y humedad DHT22

Este es un sensor de temperatura y humedad de bajo coste. El dispositivo usa un sensor capacitivo de humedad y un termistor, que devuelve una señal digital por lo que no es necesario hacer el uso de pines analógicos. El sensor se puede encontrar en un formato comercial Standard

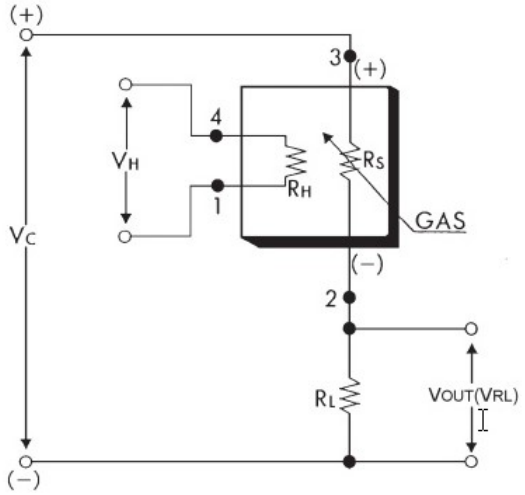



Figura 3.12: Montaje circuito de medida sensor TGS2600. Imagen tomada de [6]

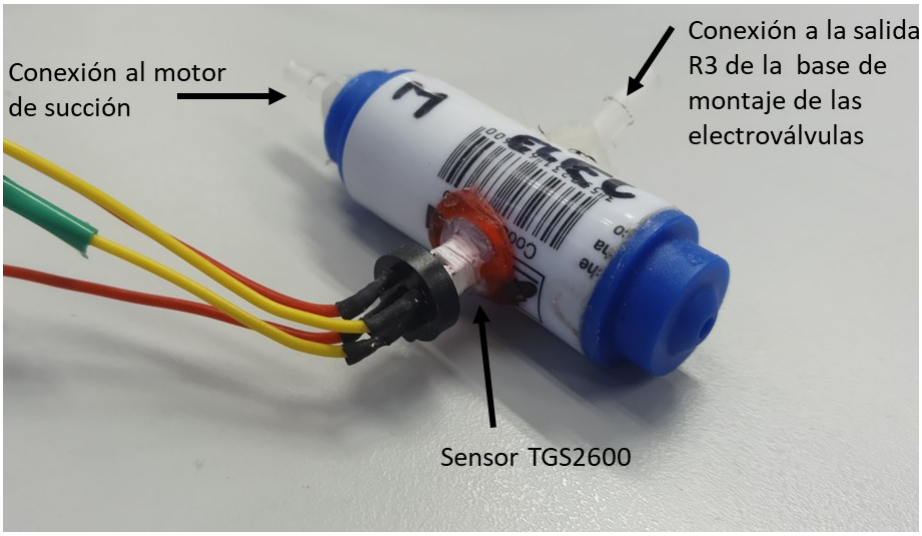



Figura 3.13: Cápsula del sensor TGS2600.

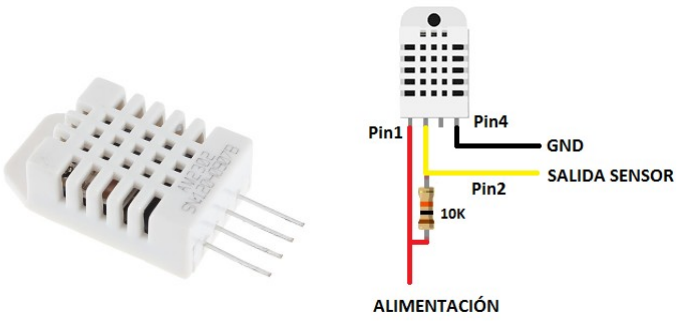
el cual viene en funda plástica y cuatro pines de conexión como se oberva en la figura 3.14(a). Este sensor es la versión mejorada de sus antecesor el DHT11, el cual ha sido usado en trabajos del GNB como en [8]. Algunas de las especificaciones técnicas de este sensor se pueden encontrar en la tabla 3.7

El formato Standard de este sensor como se ha dicho posee 4 pines para su conexión de forma que el pin 1 es la alimentación, el pin 2 es la salida del sensor, el pin 3 es un pin sin conexión y el pin 4 es GND. En la figura 3.14(b) se observa el circuito de conexión del sensor DHT22 [25].

## 3.6. Módulo de acondicionamiento

Este módulo se encarga como su nombre lo dice de acondicionar las señales que entran y salen del sistema, de tal forma que los circuitos lógicos estén aislados de los circuitos de potencia como motores o electroválvulas.

En esta parte del diseño se ha basado por una parte en los circuitos diseñados en [5] los cuales son los que se han pasado al diseño en PCB, y por otra parte se diseña una placa de




(a) (b)

Figura 3.14: (a) Sensor de temperatura y humedad DHT22. (b) Circuito de conexión sensor.

| **Modelo** | DHT22 | |
|---|---|---|
| **Alimentación** | 3,3-6VDC | |
| **Señal de salida** | Señal digital con un único bus | |
| **Rango de medida** | Humedad 0-100 %RH | Temperatura -40∼120 Celsius |
| **Precisión** | Humedad ±2 %RH (Max ±5 %RH) | Temperatura ±0, 2Celsius |
| **Resolución** | Humedad 0,1 %RH | Temperatura 0,1Celsius |
| **Repetibilidad** | Humedad ±1 %RH | Temperatura ±0, 2 Celsius |
| **Histersis de humedad** | ±0, 3 %RH | |
| **Estabilidad a largo plazo** | ±0, 5 %RH/año | |
| **Periodo de sensado** | 2s | |
| **Tiempo de respuesta** | Humedad <5s | Tempertura <10s |

Tabla 3.7: Especificaciones técnicas sensor DTH22, tabla adaptada de [25].

protección para las entradas del ADC de la placa BBB.

### 3.6.1. Circuito de protección entradas ADC BBB

Este circuito es nuevo dentro de la placa de experimentación y se realiza su diseño para proteger las entradas de los pines ADC de la placa BBB ya que como se ha comentado en la sección 3.2.1, esta tiene algunas restricciones con respecto a la tensión de entrada de estos pines. Para la protección de la placa hay varias opciones de circuitos que se presentan en [15] y es de donde se saca la idea para implementar este circuito.

Muchos de los sensores analógicos requieren tensiones de alimentación de entre 3,3 V o 5 V, lo que hace que surja la preocupación en el momento de realizar el diseño del circuito de interfaz entre el sensor y la placa. Aunque las entradas del ADC de la BBB tiene un circuito interno de protección a la entrada, estos no están diseñados para soportar corrientes durante largos periodos de tiempo. Estos circuitos eléctricos esta diseñados principalmente para la protección de descarga electro-estática (ESD). Por tanto, es necesario tener un circuito de protección externo para evitar daños sobre la placa. A continuación, se explicarán dos posibles circuitos que pueden ser usados como circuito de protección para la BBB.

#### 3.6.1.1. Diodos de protección

Los diodos de protección son un circuito simple que es usado típicamente para limitar la tensión aplicada a los pines ADC [15]. El circuito como se muestra en la figura 3.15, se basa en el uso de dos diodos y una resistencia que trabaja como limitador de corriente. El funcionamiento del circuito es el siguiente, los diodos están polarizados en inversa cuando la tensión está dentro del rango de 0 V y la $V_{ref}$ y por tanto la corriente circula a través de la entrada AINX. Sin embargo, cuando la tensión de entrada $V_{IN}$ excede los límites de $V_{ref}$ (mas la caída de tensión del diodo), el diodo superior quedará polarizado en directa y por tanto la corriente seguirá el camino hacia $V_{ref}$. Se debe tener en cuenta que si se usan diodos de silicio estos en directa tienen una caída de tensión de 0,7 V aproximadamente y si se suma a esto la $V_{ref}$ que es 1,8 V, la tensión a la cual funcionara el diodo superior será de hasta 2,5 V. Por otra parte, el diodo inferior funcionara cuando Vin está entre 0 V y 0,7 V, en este caso el diodo entrara a trabajar en directa y AGND generara corriente. La resistencia puede ser usada para limitar el nivel de corriente superior.

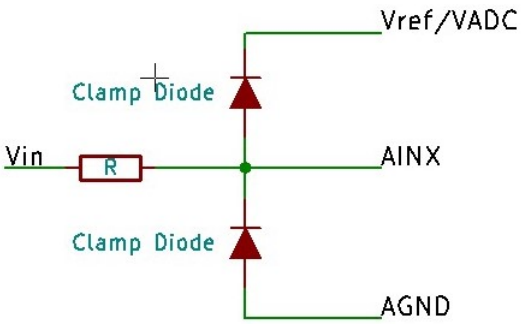



Figura 3.15: Circuito con diodos de protección, figura adaptada de [15].

Tras explicar el funcionamiento de este circuito de protección puede apreciarse que este no es el más adecuado para usarse con la BBB. Uno de los motivos por los que no se recomienda este circuito con la BBB es que la tensión en directa que cae sobre un diodo de silicio hace que el limite tensión sea mayor, en este caso entre -0,7 a 2,5 V, lo cual está fuera de los niveles de tensión de entrada de los pines ADC de la BBB.

#### 3.6.1.2. Circuito de protección con amplificadores operacionales

Aunque este circuito parece complejo figura 3.17, solo es necesario usar 3 circuitos integrados (CIs) y 7 resistencias para proteger cada uno de los pines ADC (AINX). El circuito consiste en un seguidor de tensión con amplificador operacional figura 3.16, el cual va a proporcionar a la salida la misma tensión que se aplica a la entrada Èste circuito es útil, ya que la impedancia que hay a la entrada del operacional es alta lo que hace que haya un efecto de aislamiento a la salida con respecto a la señal de entrada, de esta forma se anulan los efectos de la carga.

Por tanto, el circuito de protección en una primera etapa usa un seguidor de tensión el cual esta alimentado a 5 V y provee una tensión de salida de 1,8 V que es la tensión de alimentación que usan los otros amplificadores. Por ende, a la entrada de este primer amplificador siempre habrán 1,8 V que provienen de uno de los pines de la placa BBB ($P9\_32$).

El otro grupo de amplificadores se configura como un seguidor de tensión y son alimentados con una tensión 1,8 V. La patilla VDD del amplificador se conecta a los 1,8 V y la patilla VSS se conectado a GND, de esta forma se consigue que en la salida no se excedan los niveles de tensión de los pines ADC, entre 0 V y 1,8 V. Por último, la resistencia de $100K\Omega$ asegura que la

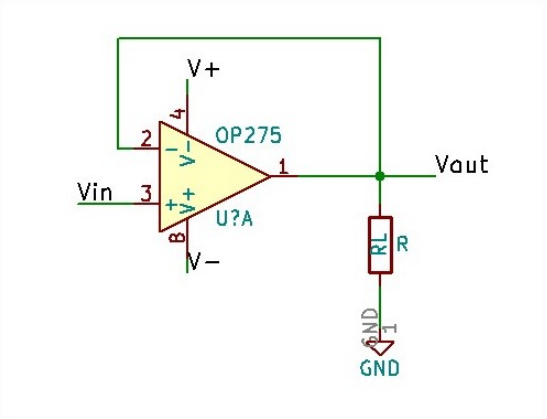



Figura 3.16: Circuito seguidor de tensión.

corriente fluya por GND a través de la resistencia en vez de que esta corriente fluya por AINX. Este es el circuito es recomendado para proteger los pines ADC de la BBB [15].

### 3.6.2. Diseño circuito de protección en PCB

El circuito de protección que se ha decido usar ha sido el basado en amplificadores operacionales, siguiendo el diseño que se presenta en [15]. En el anexo C se puede encontrar el esquema de este circuito.

El circuito que se ha construido es el que se puede apreciar en la figura 3.18, y en la tabla 3.8 se detalla una descripción de los componentes de la placa.

| | Componente | Descripción |
|---|---|---|
| **1** | Conector 1X8 | Puerto de conexión entradas analogicas (AIN0 ..AIN6) |
| **2** | MCP6002 | AO en modo seguidor de tension, a su salida se obtiene $V_{ref}$ =1,8 V |
| **3** | MCP6004 | AO se encarga que las entradas analógicas nunca superen los rangos de tensión de entrada del ADC |
| **4** | Conector 2x6 | Puerto de conexión de las salidas analogicas hacia la placa BBB |

Tabla 3.8: Descripción de componentes circuito de protección, ver figura 3.18.

## 3.7. Circuito placa experimentación

Una de las ideas claves de este proyecto ha sido poder tener todos los circuitos desarrollados en [5] dentro de una sola placa PCB, la cual se pueda adaptar fácilmente y que sea versátil. Por tanto, en esta parte se explicara el diseño del PCB además de los cambios hardware que se han realizado con respecto al diseño en protoboard.

### 3.7.1. Circuitos implementados en la placa de experimentación

La placa de experimentación desarrollada en [5], cuenta con diferentes circuitos usadas para el control de los diferentes componentes así como para el uso de técnicas de modulación. A

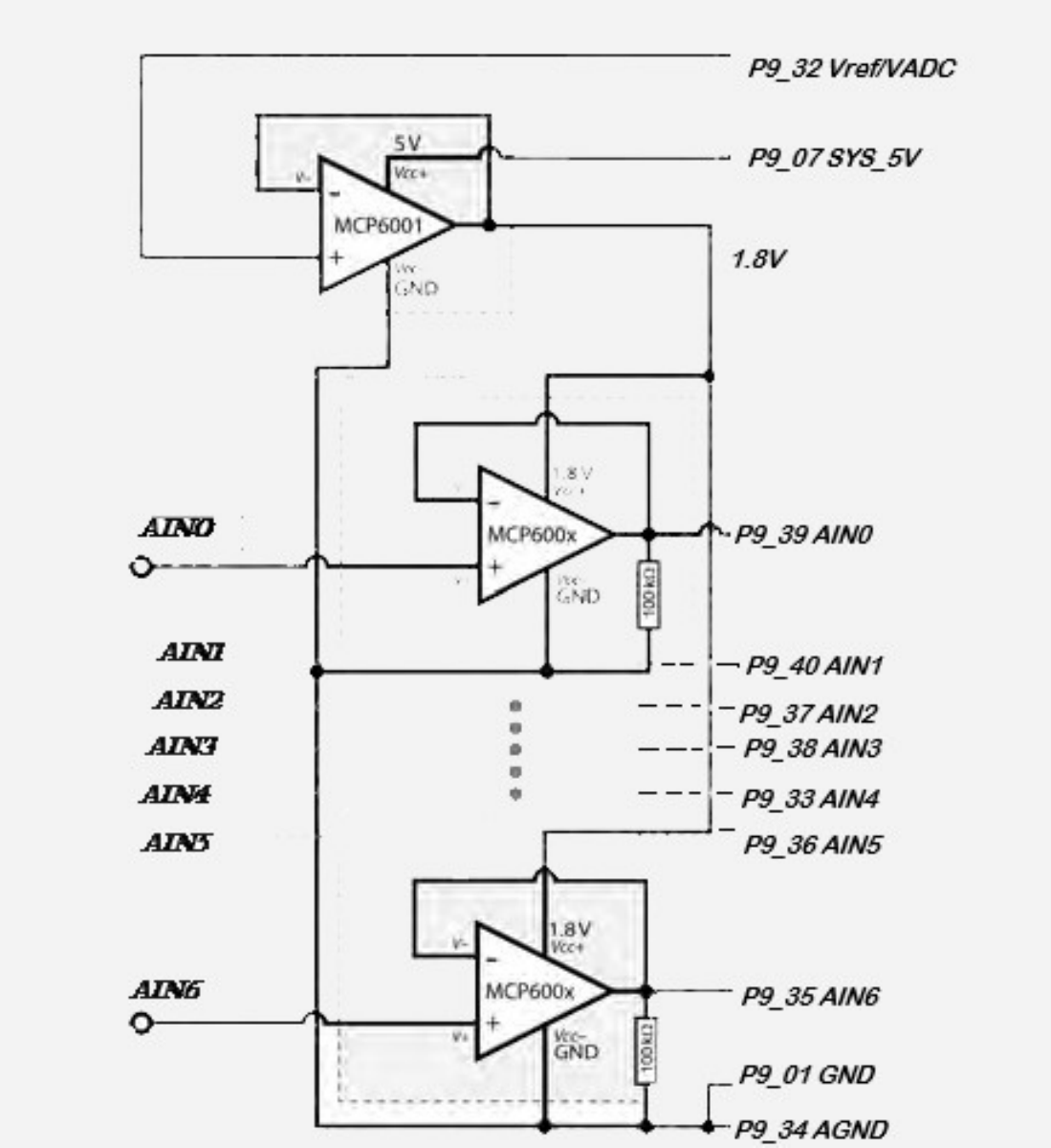



Figura 3.17: Protección entradas ADC de BBB usando AO, figura adaptada de [15]

continuación, se explicara de forma breve los circuitos que se han integrado en la placa de experimentación.

#### 3.7.1.1. Circuito para la modulación en frecuencia

El circuito de modulación en frecuencia se basa en el uso de un multivibrador de tipo 555. Para la realización de este proyecto se ha usado un NE555 [26], la implementación del circuito parte del diseño realizado en [5] en donde se explican la elección de los diferentes valores de los componentes que se han usado. En la figura 3.19, se muestra el esquema del circuito usado para modulación en frecuencia con sus respectivas entadas salidas a la BBB. Como se ha explicado en la sección 3.2.1.1, la BBB tiene ciertas limitaciones que se deben tener en cuenta cuando se usan los pines de entrada o los de salida. En este caso se usará el pin P9_12 el cual es un pin GPIO ver anexo F, y este tiene su valor máximo a 3,3 V es por este motivo que dentro del esquema se usa un circuito de adaptación para poder limitar la tensión. En este caso en [5] se decidió usar

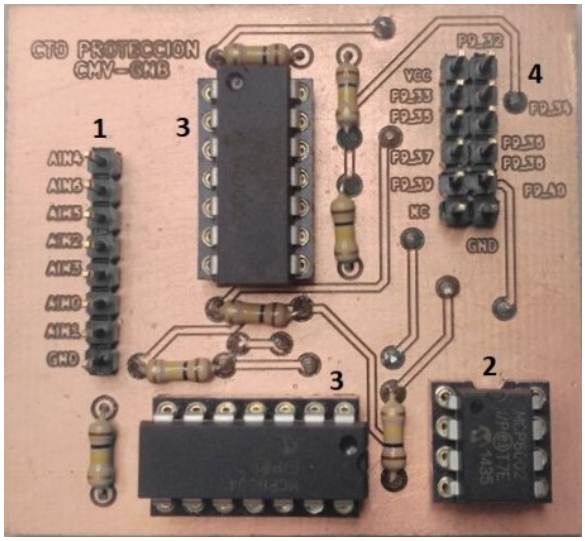



Figura 3.18: PCB circuito de protección para pines ADC de la BBB

transistores Darlington.

El componente usado para adaptar el circuito y evitar daños en la BBB ha sido el integrado ULN2003 [27], el cual permite limitar la tensión al voltaje que se desee. De tal forma que si se fija la tensión de alimentación de este componente a 3,3 V, se puede asegurar que la corriente y la tensión de entrada a la BBB no va a superar los máximos como se describe en la tabla 3.2.

#### 3.7.1.2. Circuito para la modulación en amplitud

El circuito usado para la modulación en amplitud fue inicialmente desarrollado en [4], y se diferencia principalmente del circuito diseñado en [5] en que en el primero se usaba el sensor TGS2611 [28] y en el segundo el sensor TGS2600 [6]. En la figura 3.20, se muestra el esquema del circuito que ha sido usado en [5] para la plataforma de experimentación. Al igual que en el circuito de la sección 3.7.1.1, en este también se realizaron las adaptaciones necesarias para limitar la tensión de salida y así no estropear la BBB, en este caso en [5] se decidió usar un divisor de tensión.

Por otra parte como se ha mencionado en la sección 3.2.1.1, los pines digitales configurados como PWM pueden entregar tensiones entre 0 V y 3,3 V. El calentador del TGS2600 [6] funciona con tensiones entre 0-5 V, y la salida del PWM solo es capaz de dar hasta 3,3 V. Por tanto, en [4] se implemento un circuito para ampliar el rango dinámico del calefactor consiguiendo hasta 5 V, que se corresponde con la tensión máxima de calentamiento. Este circuito esta basado en el uso de un transistor NPN PN222A [29]. La elección de usar un transistor NPN es debido a que el control de la base del transistor se lleva a cabo mediante el pin PWM de la BBB, el cual esta limitado a 3,3 V. Como se quiere tener una relación directa entre el PWM y la tensión del calefactor a la vez que exista un punto de corte coincidente a 0 V, se elige un transistor NPN.

El funcionamiento del circuito de adaptación es el siguiente:

- Cuando se aplican 0 V en la salida del PWM el transistor no permite el paso de la corriente al colector.
- Cuando se aplica tensión por el pin PWM, en la base del transistor este dejara el paso de corriente y se conseguirá crea una diferencia de tensión entre el emisor y el colector que sera la tensión de calentamiento.

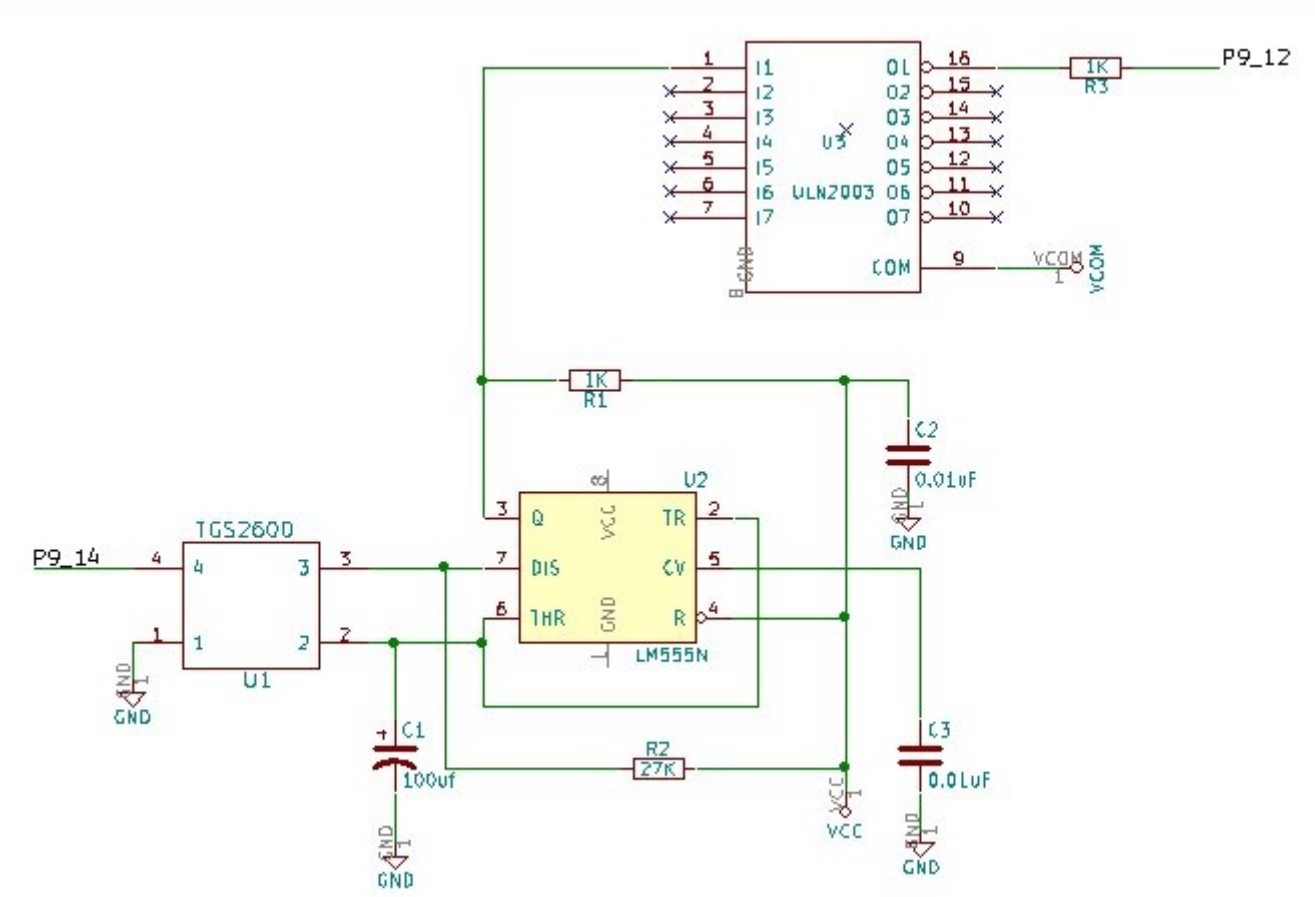



Figura 3.19: Esquema del circuito multivibrador con sensor TGS2600 [6] y NE555 [26], usado para la modulación en frecuencia.

El circuito de modulación en amplitud basa su funcionamiento en variar la tensión del calefactor del sensor a través de una señal PWM, por lo que a su vez esta tensión de calentamiento también se podría dejar fija con lo que el mismo circuito usado para la amplitud puede ser usado como circuito de modulación pura. Por tanto, el circuito para la modulación en amplitud puede usarse también como circuito sin modulación, ya que por software se puede controlar que el ciclo PWM siempre este al 100 % lo que equivale a alimentar el sensor 5 V durante el periodo de la señal PWM. Esta es una de las mejores que se ha realizado respecto al diseño realizado en [5], en el que había un circuito solo para el sensor TGS2600 [6] sin usar ninguna modulación.

#### 3.7.1.3. Circuito de control de electroválvulas

Para la desarrollo de este proyecto y como se describe en la sección 3.3.1, se han usado 4 y 5 electroválvulas SY114-5LOZ [23] respectivamente para las plataformas construidas. Estas electroválvulas trabajan con una tensión de 24 V, por lo que es necesario usar una fuente de alimentación ya que la BBB puede dar hasta 5 V. Por tanto, se usa una fuente de alimentación capaz de dar a la salida 24 V y una corriente de 450 mA.

El control de las electroválvulas se lleva a cabo usando un CI ULN2003 [27], el cual como se ha mencionado ya en la sección 3.7.1.1 es un array de transistores Darlington y comúnmente es usado para la conmutación.

El esquema del circuito de electroválvulas se puede ver en la figura 3.21, en este caso el ULN2003 [27] es alimentado con una tensión de 24 V en el pin 9, que proviene de la fuente de

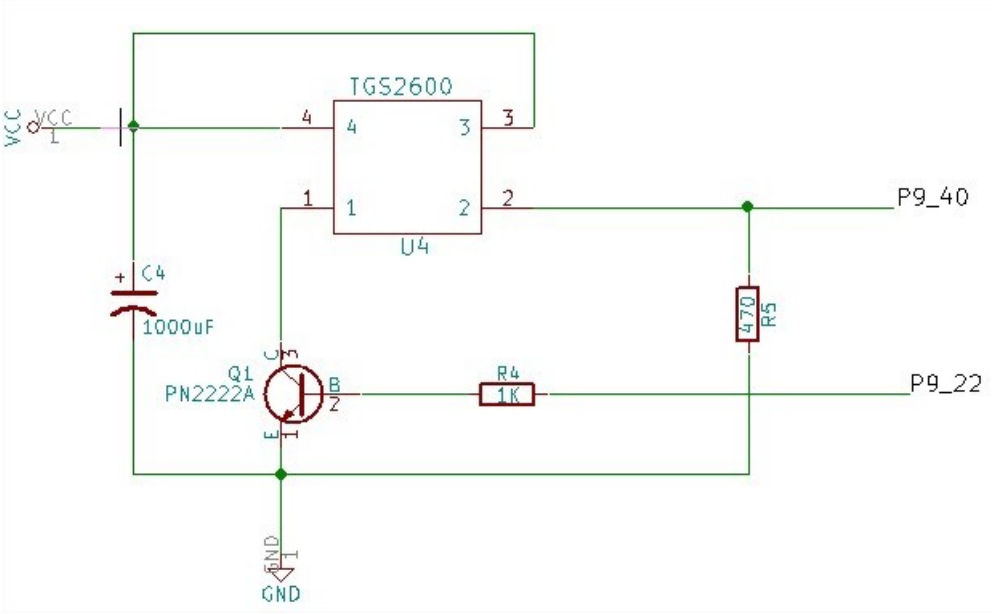



Figura 3.20: Esquema del circuito de amplitud.

alimentación. Cada pin de entrada del CI se conecta a una resistencia de protección $1K\Omega$ que viene de los pines de salida del a BBB y que serán los encargado del control de las electroválvulas. Los pines de salida del CI se conectaran al polo negativo de cada electroválvula y el polo positivo de cada electroválvula estará conectada a 24 V. El funcionamiento de este circuito es el siguiente: cuando desde la BBB se envíe un pulso a nivel alto por alguna de las salidas que se configuren para el control de electroválvulas, el CI ULN2003 [27] conmuta y permite el paso de corriente por la electroválvula que se haya activado.

#### 3.7.1.4. Circuito de potencia del motor

Los motores elegidos para el proyecto han sido el D250-BL y 220-BL [24], usados en la primera y segunda plataforma respectivamente, como se ha explicado en la sección 3.3.3. El motor funciona a diferentes potencias de succión dependiendo de la tensión de alimentación que se aplique y esta puede variar entre 1,5 a 5 V.

Como la BBB no tiene puertos PWM que permita variar su valor entre las tensiones que necesita el motor, en [5] fue necesario crear un circuito de adaptación. La salida PWM de la BBB da una tensión de 3,3 V y 0 V como valores máximo y mínimo respectivamente. El circuito de adaptación usa un TIP120 [30], el cual es un transistor que funciona como amplificador o interruptor electrónico. El funcionamiento del circuito se lleva cabo cuando se envía una señal a nivel alto a la base del transistor. El transistor cambia y permite que la corriente fluya desde el colector hacia el emisor. Por tanto, en función del valor de señal PWM que se envíe desde la BBB se dejará pasar más o menos corriente entre el colector y el emisor del TIP120 [30].

Por ultimo, es importante tener en cuenta y como se detalla en [5] al conectar la alimentación del motor se produce un pico negativo de tensión que puede dañar la placa. Por tanto en el diseño en [5] se usa un diodo como rectificador. Este diodo funcionara como una válvula unidireccional que solo va a permitir el flujo de corriente en una solo sentido, el diodo usado es el 1N4001 [31]. Como ultima medida de protección se usa una resistencia de $1K\Omega$ para proteger la base del transistor en caso de que se produzca un exceso de corriente que podría dañarlo.

Por ultimo, en el diseño en [5] se hacia uso de una sola fuente de alimentación externa ya

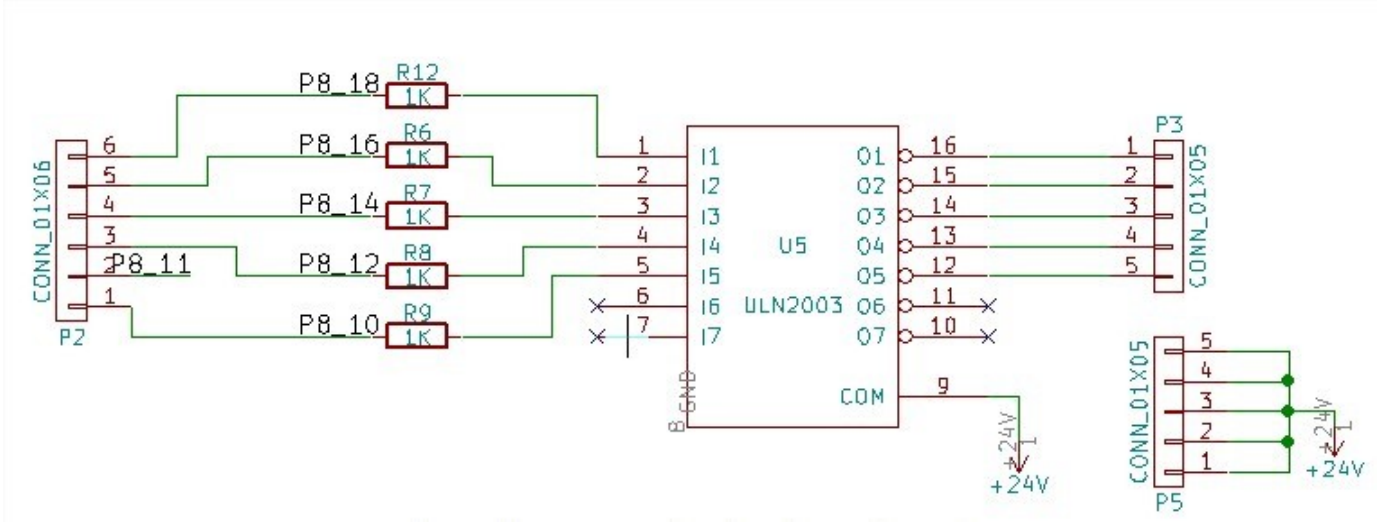



Figura 3.21: Esquema del circuito de electroválvulas.

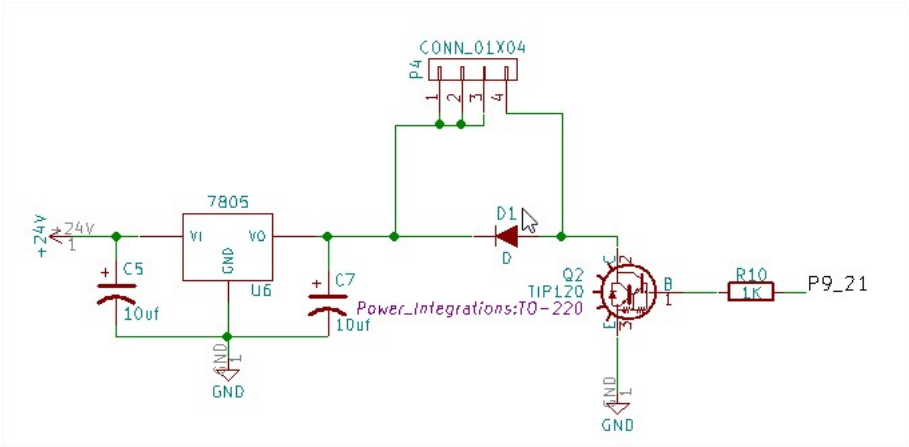



Figura 3.22: Esquema circuito de acondicionamiento para el motor, integrado en la figura 3.24 y 3.25.

que las electroválvulas se alimentaban con una tensión de 6 V, esto hacia que la misma tensión que se usaba para las electroválvulas se pudiese usar para alimentar el motor de forma directa y por ende solo se usaban una alimentación externa para el circuito.

En este nuevo diseño a simple vista se debería hacer uso de dos alimentaciones externas una para las electroválvulas y otra para el motor. Para evitar el uso de dos fuentes externas lo que se hace es introducir un regulador de tensión el cual se pone justo antes de la entrada de tensión del motor como se muestra en la figura 3.22. El regulador usado ha sido el modelo R-78E5 del fabricante recom [32], el cual tiene una tensión de entrada de 7 a 28 V y a la salida se podrán obtener 5 V, con una corriente de salida de 1 A. El uso de este regulador es sencillo ya que sus pin-out es compatible con los de un regulador LM78XX, que es uno de los reguladores más comunes en el mercado. Con la integración del regulador se consigue solucionar el uso de una segunda fuente de tensión externa para alimentar el motor.

#### 3.7.1.5. Circuito de adquisición de temperatura y humedad

El circuito de temperatura y humedad es un circuito bastante sencillo de implementar como se explica en la sección 3.5 y como se puede observar en la figura 3.14(b). En la figura 3.23, se

muestra el esquema del circuito.

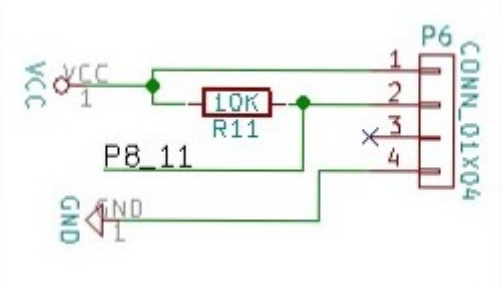



Figura 3.23: Esquema circuito de adquisición de temperatura y humedad.

### 3.7.2. Construcción placa PCB

Con los cambios explicados sobre el diseño original y los nuevos cambios aplicados, es importante destacar que se han construido dos placas de experimentación. La primera placa construida figura 3.24, adapta los nuevos cambios realizados y esta preparada para usar tan solo 4 electroválvulas.

La segunda versión de placa de experimentación figura 3.25 es una mejora de la primera versión, en este caso la placa está preparada para poder llevar a cabo el control de las 5 electroválvulas, además de que la resistencia de carga que se usa en el circuito de amplitud puede cambiarse, puesto que como se menciona en 3.3.1 esta resistencia sirve para llevar a cabo la medida de tensión del circuito por tanto es un valor que se puede cambiar en cualquier momento.

En los anexos D y E se pueden encontrar los esquemáticos de la primera y segunda plataforma de experimentación respectivamente así como sus diseños en PCB.

Al finalizar este capítulo se consigue la construcción de dos placas de experimentación. La primera placa ha sido usada para las diferentes pruebas y comprobar que el diseño era correcto. Esta placa esta diseñada para poder hacer pruebas hasta con 4 odorantes, lo que conlleva que se han usado 4 electroválvulas. La segunda placa ha sido el diseño final de la plataforma de experimentación en la cual se han hecho las mejoras de la primera plataforma y además en esta placa se pueden usar hasta 5 odorantes para los experimentos, por tanto se ha agregado una electroválvula demás con respecto a la primera versión.

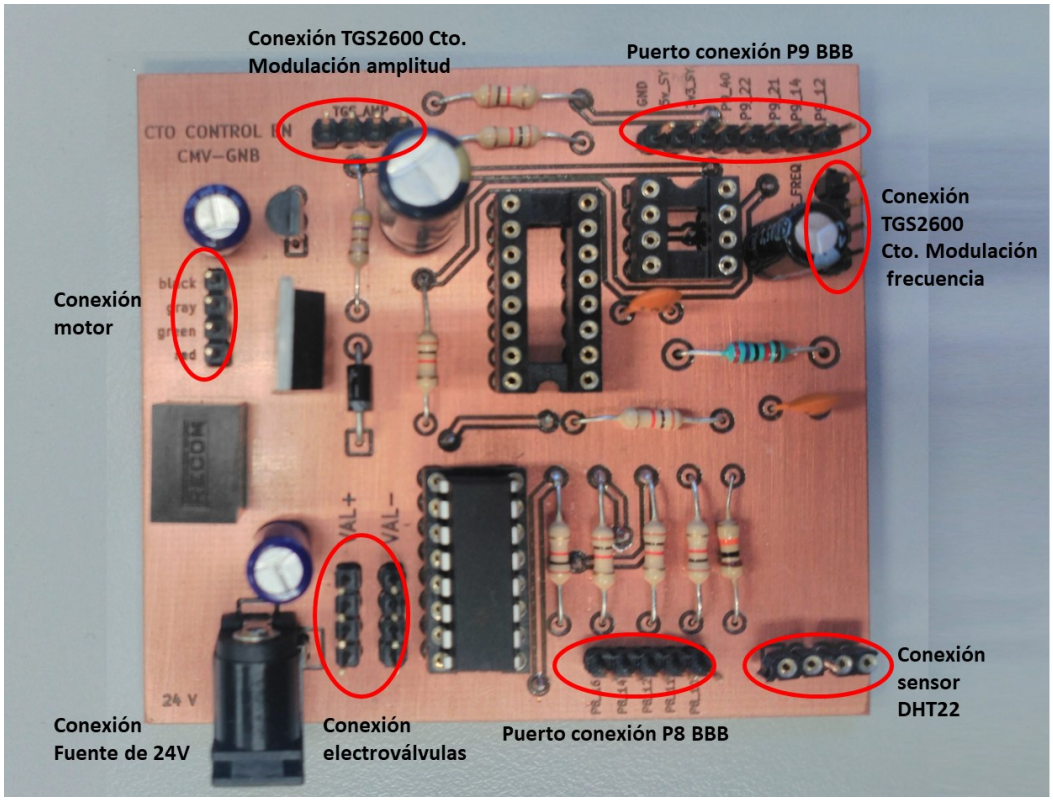



Figura 3.24: Primera plataforma de experimentación.

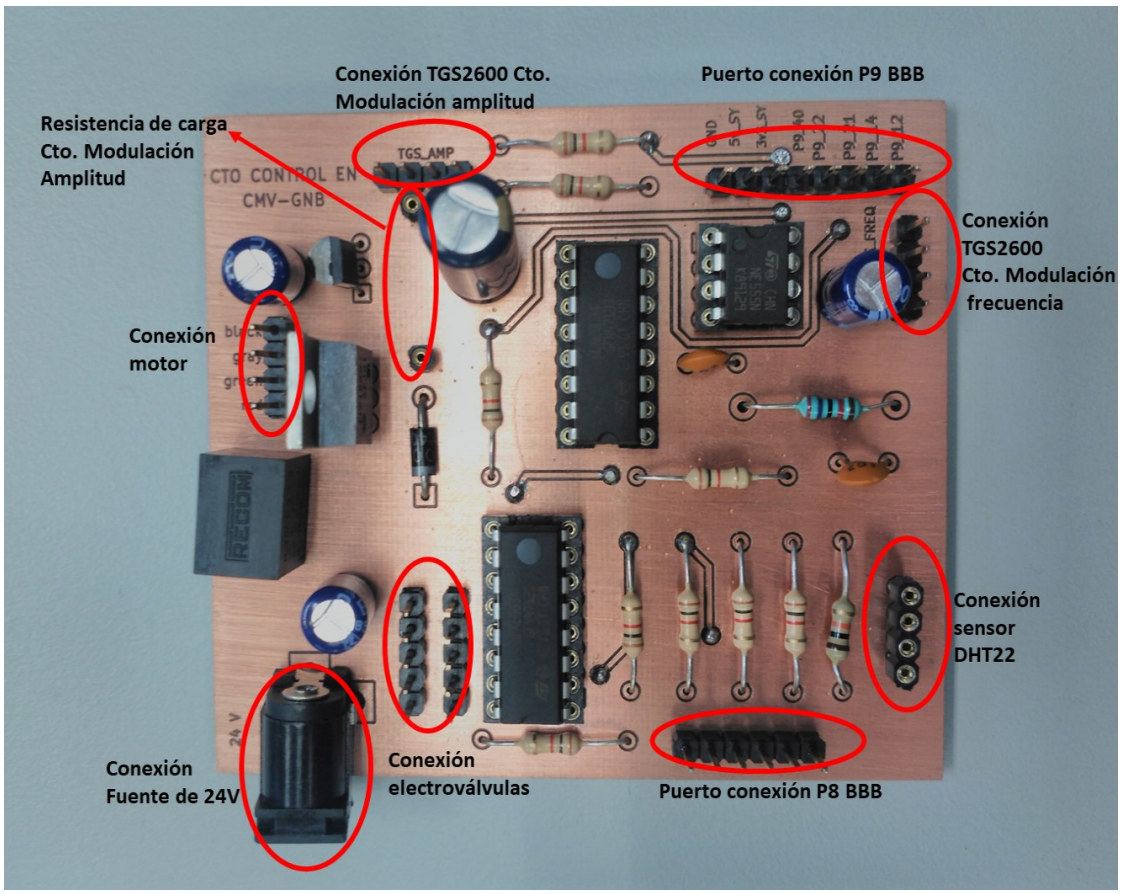



Figura 3.25: Segunda plataforma de experimentación.

# 4

# Funcionamiento de las plataforma y pruebas

## 4.1. Introducción

En el capítulo 3 se ha explicado de forma breve los componentes que posee la placa, así como el diseño que se ha realizado partiendo del diseño en protoboard que fue montado en [5].

El objetivo de este capítulo es dar las nociones básicas del funcionamiento de la placa a nivel de usuario y por otra parte comprobar el buen funcionamiento de la placa para los circuitos de las modulaciones en amplitud y frecuencia así como el funcionamiento de la placa de protección. Las pruebas que se han realizado para los circuitos de amplitud y frecuencia, se han llevado a cabo usando el software desarrollado en [33], el cual implementa un lenguaje especifico para una plataforma de adquisición de odorantes como la que se ha desarrollado en el proyecto. El uso de este software ha permitido que las pruebas sean más versátiles en el momento de realizar los experimentos.

También es importante destacar que en este capítulo se hará un estudio breve del comportamiento del sensor ante las muestras de odorantes bajo estudio, ya que como se ha mencionado en 3.4.1, el comportamiento del sensor TGS2600 no es el mismo siempre y por tanto será necesario primero ver cómo reacciona ante las muestras que se estaban usado en [5] y ver si estas muestras serán válidas para el desarrollo del proyecto.

Antes de empezar a usar la BBB se recomienda leer el anexo B, en el cual se encuentran todos los pasos de configuración de la BBB.

## 4.2. Circuito de protección

Para el circuito de protección que se explico en la sección 3.6.1, se comprobará que a la salida haya como máximo una tensión 1,8 V cuando la tensión de entrada es mayor a este valor. Para ello se puede usar la BBB y una fuente de alimentación. En la figura 4.1, se señala todos los pines que son necesarios conectar para el funcionamiento de este circuito y en el anexo C se puede encontrar el esquemático de este circuito. El procedimiento a seguir es el siguiente:

1. Lo primero que se debe comprobar es que la tensión de alimentación del circuito llega a cada uno de los CIs que forman la placa. Aunque en la figura 4.1, se muestra los CIs

colocados sobre la placa, siempre las pruebas de alimentación se deben hacer sin los CIs para así evitar que se estropeen en caso de que haya algún corto circuito. El circuito de protección se debe conectar a una tensión de alimentación de 5 V que va conectado al pin de VCC de la placa, en este caso esta tensión de alimentación viene desde la BBB y se pueden usar los pines P9_05, P9_06, P9_07 o P9_08 que proporciona una tensión de 5 V, ver anexo F. A continuación se conecta la placa a GND desde la BBB usando el pin P9_34, como se observa la placa tiene varios pines de conexión a GND con conectar uno de ellos sera suficiente. Una vez conectada la alimentación se comprueba que al CI MPC6002 [34] en el pin 8 le llegan 5 V.

2. Una vez comprobado que el MCP6002 [34] esta siendo alimentado correctamente, se pasa comprobar que este CI a su salida funciona como un seguidor de tensión, ya que a su entrada se recibe una tensión de 1,8 V que es la tensión máxima que podrá llegar a un pin del ADC en la BBB. En este punto se asume que ya estará sobre le zócalo el CI MCP6002 [34]. A la placa primero se conecta desde la BBB el pin P9_32 el cual da una tensión de 1,8 V. Se comprobara que en el pin 3 del MCP6002 [34] llegan 1,8 V y que en los pines 1 y 2 también hay 1,8 V de esta forma se comprueba que el MCP6002 [34] funciona como el circuito seguidor de tensión.

3. Ahora sobre los zócalos donde van los CIs MCP6004 [34] en el pin 4 se comprueba que en este punto hay una tensión de 1,8 V que es la tensión de alimentación de estos CIs. De esta forma las señales que lleguen a estos circuitos y sean superiores a esta tensión serán capadas dando como máximo la tensión de alimentación. Si la señal de entrada esta por debajo de 0 V a la salida se tendrá 0 V ya que estos amplificadores operacionales trabajan con una única tensión de alimentación y además son rail to rail lo que hace que la tensión a la salida sea muy cercana a la tensión de alimentación de esta forma se consigue que el amplificador no se sature antes.

4. Por ultimo una vez comprobado los pasos anteriores ya se pueden poner los CIs que quedaban aun por poner en sus respectivos zócalos. Ahora se puede conectar a cualquiera de las entradas analógicas a una tensión de alimentación y probar que para tensiones entre 0 V y 1,8 V a la salida se tendrá la misma tensión que a la entrada, si la tensión es superior a 1,8 V a la salida se tendrá una tensión de 1,8 V y si la tensión de entrada es menor de 0 V a la salida se obtendrá 0 V. En la tabla 4.1, se relacionan los pines de entrada analógicos con los pines de entada analógicos de la BBB para realizar las comprobaciones necesarios sobre cada uno de los pines. Una vez comprobado este paso las "Entradas analógicas a la BBB" de la figura 4.1 ya se pueden conectar a los pines del ADC de la BBB.

| **Entradas analógicas externas** | **Entradas analógicas a la BBB** |
|---|---|
| AIN0 | P9_39 |
| AIN1 | P9_40 |
| AIN2 | P9_37 |
| AIN3 | P9_38 |
| AIN4 | P9_33 |
| AIN5 | P9_36 |
| AIN6 | P9_35 |

Tabla 4.1: Correspondencia de entradas analógicas del circuito de protección con las pines de la BBB.

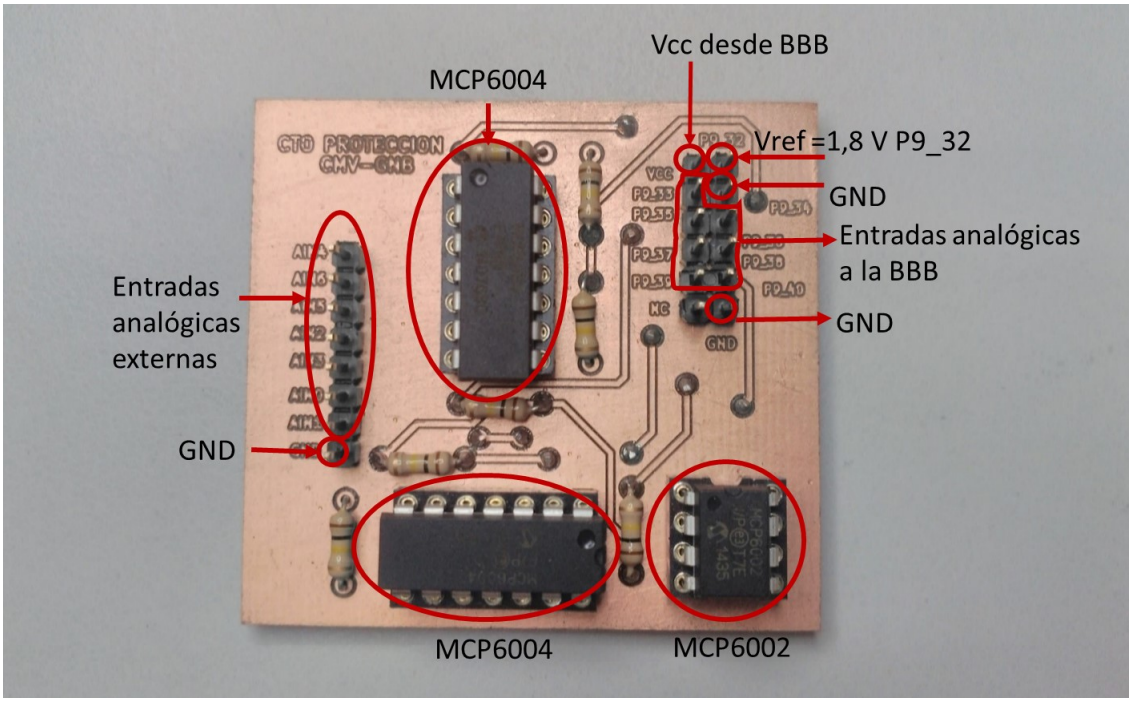



Figura 4.1: Placa de protección esquema para conexión.

## 4.3. Circuito de alimentación

En esta primera etapa lo primero que se debe hacer es comprobar que el circuito esta alimentado correctamente, en este caso se comprobará la alimentación que viene de la fuente externa y la alimentación que viene desde la BBB. Antes de hacer las comprobaciones se recomienda previamente revisar el circuito para comprobar que no hayan cortos circuitos sobre la placa y por otra parte no poner los CIs sobre su respectiva zócalo (como son el LM555 [26] y el ULN2003 [27]) para así evitar posibles daños sobre estos.

### 4.3.1. Alimentación externa

A continuación se describen los pasos a seguir para la conexión de la alimentación externa, así como comprobar las tensiones que llegan a ciertos elementos de la placa como se muestra en la figura 4.2.

1. Para la placa experimentación lo primero que haremos es conectar la fuente de alimentación de 24 V al conector correspondiente.
2. Comprobar sobre la conexión de las electroválvulas en el signo marcado con el símbolo como (+) que llegan aproximadamente 24 V, puede pasar que al momento de medir la tensión en este punto haya más de 24 V. Esto puede pasar porque la fuente de alimentación da 24 V cuando esta la carga conectada, en este caso las electroválvulas no se encuentran conectadas aún y por eso se puede tener un tensión superior.
3. Comprobar sobre el ULN2003 [27] en el pin 9 que llegan 24 V, ya que esta es la alimentación de este CI.
4. Comprobar que en el regulador 78E05 [32] en la patilla 1 llegan 24 V y en la patilla 3 se obtiene 5 V y una corriente de 500 mA, que sera la alimentación y corriente necesaria para el funcionamiento del motor como explica en la sección 3.3.3.

5. En la conexión del motor comprobar que los pines sobre la placa que pone red, green y grey llegan 5 V que provienen de la salida del 78E05 [32].

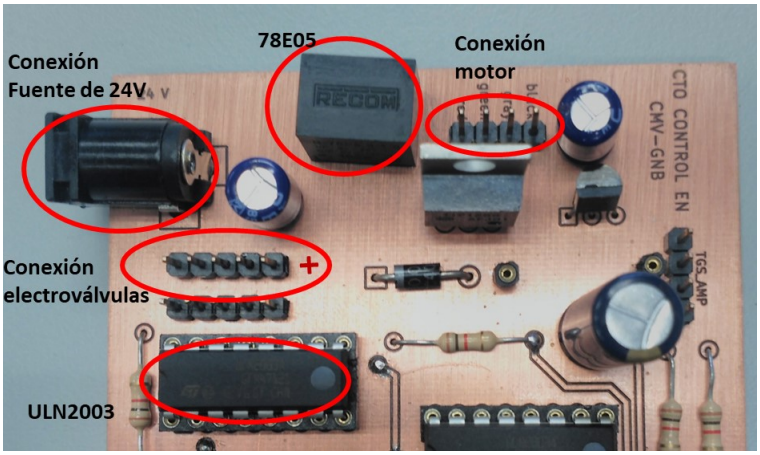



Figura 4.2: Placa experimentación conexión elementos conectados a la alimentación externa.

### 4.3.2. Alimentación desde la BBB

La placa construida además de recibir alimentación por parte de la fuente externa, también recibe tensiones de alimentación que vienen desde la BBB. Estas tensiones son usadas para alimentar el CI LM555 [26], el sensor DHT22 [25] y al TGS2600 [6] usado para la modulación en amplitud proporcionando una tensión de 5 V y el CI ULN2003 [27] al que se le proporciona una tensión de 3,3 V. A continuación se detallan los pasos a seguir para alimentar la placa con los pines de la BBB así como los pines de los CI y sensores que se deben comprobar para ver que estén correctamente alimentados como se muestra en la figura 4.1.

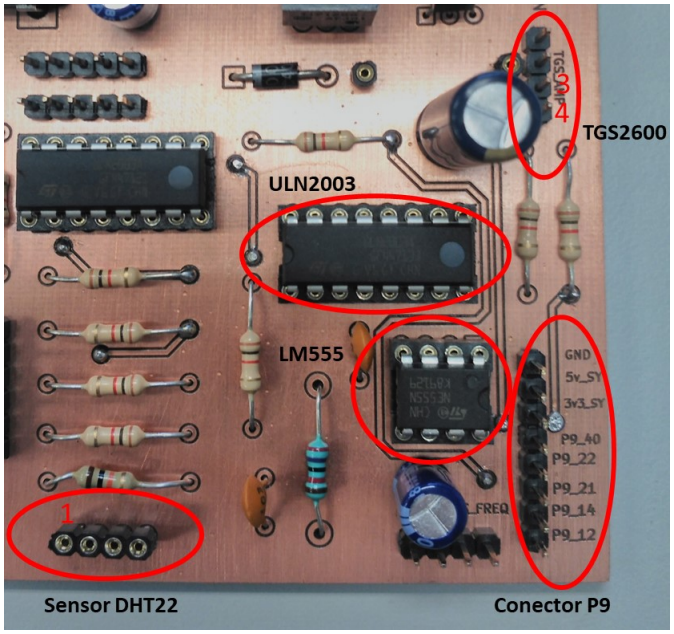



Figura 4.3: Placa experimentación conexión elementos conectados a la alimentación proveniente de la BBB.

1. Lo primero que se debe hacer es alimentar la BBB conectándola a través de la alimentación externa con la fuente de 5 V.

2. Una vez conectada la BBB comprobar que los pines que están diseñados para suministrar tensión, ver anexo F, en estos se obtienen las tensiones de 5 V y 3,3 V.

3. Una vez comprobado que se suministran las tensiones se conectaran estos pines a placa. Para la tensión de 5 V se puede coger por ejemplo el pin P9_05 de la BBB y se conectara al pin de la placa etiquetado como "5v_SY" en el conector correspondiente para los pines de expansión P9. Así mismo, se puede coger el pin P9_03 de la BBB el cual suministra 3,3 V y conectarlo al pin etiquetado en la placa como "3v3_SY" en el conector correspondiente para los pines que provienen del puerto de expansión P9.

4. El siguiente paso sera comprobar que estas tensiones llegan hasta los diferentes circuitos a los cuales se les suministran estas tensiones de alimentación.

   *a*) Se comprobara que para el sensor DHT22 [25] en el pin 1 llegan 5 V.

   *b*) En el sensor TGS2600 [6] se comprobara que a los pines 3 y 4 le llegan 5 V.

   *c*) En el CI LM555 [26] se debe comprobar que al pin 4 le llegan 5V.

   *d*) En el CI ULN2003 [27] en el pin 9 deben llegar 3,3 V.

Comprobado los pasos anteriores se podrá apagar la BBB para así poder montar los CIs en sus correspondientes zócalos y poder comenzar a usar la placa construida y comprobar su correcto funcionamiento.

## 4.4. Circuito de medida de temperatura y humedad

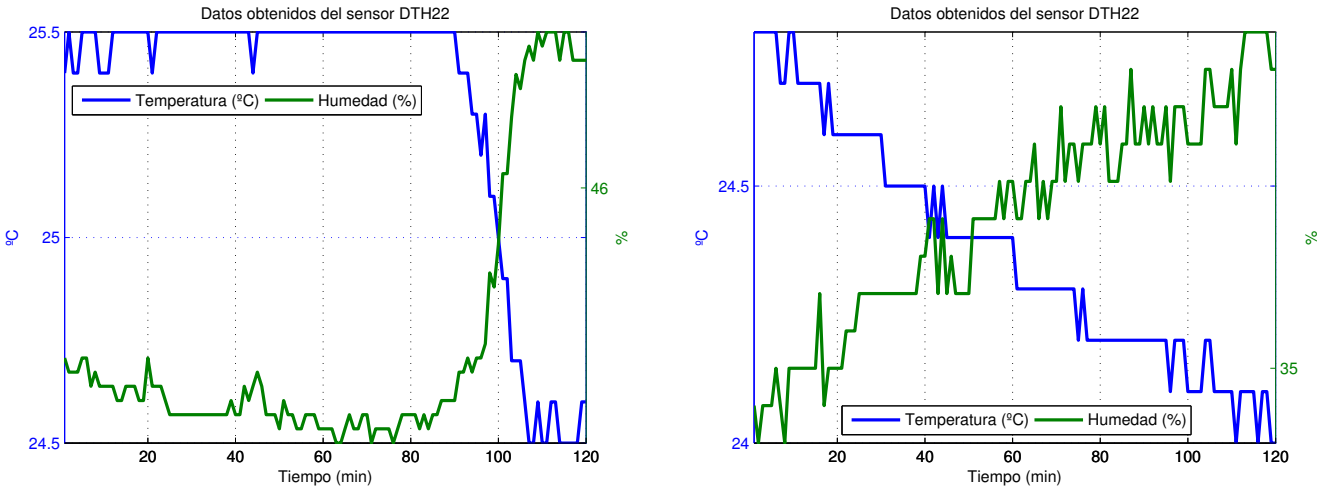
Para la prueba de este circuito será necesario tener en cuenta que solo se usa uno de los pines de las BBB, en este caso el pin usado es el P8_11, el cual es un pin GPIO y por tanto sera necesario tener instalada la librería Adafruit_BBIO ver anexo B.3.2, para poder llevar a cabo el proceso de lectura de datos que envía el sensor. Este sensor nos da datos de la medida de temperatura y humedad una vez por minuto para la prueba que se ha realizado. El código usado para la prueba de este circuito ha sido basado en el código desarrollado en [5], este código se puede encontrar en el anexo I.1.

Para realizar la prueba la placa BBB deberá estar conectada a la alimentación de 5 V y una vez se pueda acceder a la consola de la placa se lanzará el siguiente comando para comenzar a capturar los datos del sensor:

*python3 temperatuta.py tiempo*

En donde tiempo, será el tiempo en minutos que el sensor estará midiendo los valores de temperatura y humedad. Se han realizado dos experimentos en distintos instantes de tiempo, en los cuales el sensor ha estado midiendo la temperatura del laboratorio B208 de la Escuela Politécnica Superior de la UAM durante un tiempo de 120 minutos. Estos dos experimentos además de haberse realizado en distintos instantes de tiempo, también se diferencian en que cada uno de ellos se realiza sobre cada una de las plataformas de experimentación que se ha construido. Los resultados obtenidos se pueden observar en la figura 4.4. En la figura 4.4(a), se presenta el experimento realizado sobre la primera placa y en la figura 4.4(b) se observan los resultados obtenidos en la segunda placa. En ambos experimentos se ve como la temperatura y

la humedad no son variables en el entorno ya que se aprecia que no hay cambios bruscos. En la figura 4.5, se muestra la lectura del sensor usando la segunda placa, solo que esta vez la duración del experimento ha sido durante mas tiempo (24 horas). El sensor empezó a leer durante la noche y como se puede ver la temperatura durante los primeros 600 minutos (10 horas) se mantiene entre los 22 ºC y 23 ºC, que sería algo normal teniendo en cuenta que durante la noche la temperatura del interior del laboratorio debería descender. En los siguientes 600 minutos, se observa como la temperatura empieza ascender de manera gradual que se corresponderían con las horas del día en las que dentro del laboratorio ya hay actividad. Por último, durante los últimos 200 minutos del experimento se ve como la temperatura desciende de nuevo.




(a) (b)

Figura 4.4: Lectura obtenida del sensor DHT22 [25]. (a) Resultados obtenidos sobre la primera placa de experimentación durante un tiempo de 2 h. (b) Resultados obtenidos sobre la segunda placa de experimentación durante un tiempo de 2 h.

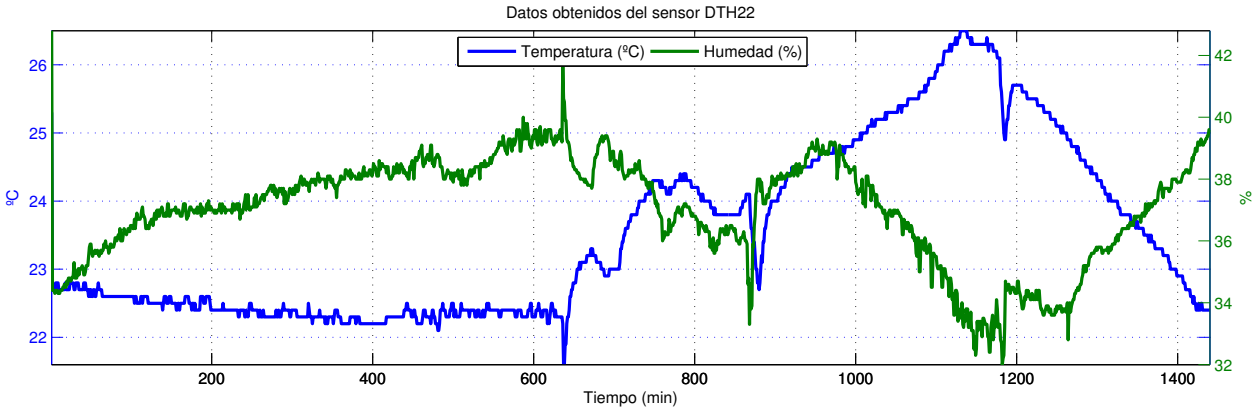



Figura 4.5: Lectura obtenida del sensor DHT22 [25] con la segunda placa de experimentación durante 24 horas.

Los datos que se obtienen de la lectura del sensor son almacenados por una parte en un fichero de texto plano ( formato "txt") el cual guarda la información de la captura como la hora de inicio, hora de fin y los valores leídos por el sensor. Por otra parte, también se almacena un fichero que contiene los datos de lectura del sensor que es usado principalmente para poder trabajar con los datos obtenidos. Es importante tener estas medidas ya que pueden influir en la respuesta que ofrezca el sensor TGS2600 [6].

## 4.5. Control del circuito de electroválvulas

Para el control de las electroválvulas se deben asignar pines de la BBB para poder controlar cada uno de las electroválvulas. Los puertos asignados para el control se puede ver en la tabla 4.2

| **Puerto** | **Nombre** | **Odorante placa 1** | **Odorante placa 2** |
|---|---|---|---|
| P8_10 | electrovalve1 | Metanol | Metanol |
| P8_12 | electrovalve2 | Etanol | Etanol |
| P8_14 | electrovalve3 | Butanol | Butanol |
| P8_16 | electrovalve4 | Aire | Aire |
| P8_18 | electrovalve5 | | Metanol |

Tabla 4.2: Pines de la BBB usados para el control de electroválvulas en las placas diseñadas.

Las electroválvulas en la realización del proyecto dejan el paso del odorante cuando están a nivel bajo. Previamente se debe configurar los puertos para que sean salidas digitales para ello a continuación se indican la configuración que se debe usar en Python.

```
import Adafruit_BBIO.GPIO as GPIO
## Asignacion de puerto
electrovalve1 = 'P8_10' #Electrovalvula1 (METANOL)
electrovalve2 = 'P8_12' #Electrovalvula 2 (ETANOL)
electrovalve3 = 'P8_14' #Electrovalvula 3 (BUTANOL)
electrovalve4 = 'P8_16' #Electrovalvula 4 (AIRE)
electrovalve5 = 'P8_18' #Electrovalvula 5 (ODORANTE ELECTROVALVULA 1)

## DECLARACION DE SALIDAS DEL SISTEMA.
GPIO.setup("P8_10",GPIO.OUT)
GPIO.setup("P8_12",GPIO.OUT)
GPIO.setup("P8_14",GPIO.OUT)
GPIO.setup("P8_16",GPIO.OUT)
GPIO.setup("P8_18",GPIO.OUT)
```

Tras la configuración de los pines se puede implementar una función que sea llamada desde el programa principal para poder controlar las electroválvulas. Esta función recibe como parámetro de entrada la electroválvula que se quiere activar en cada momento. Por tanto, la función que permite el control de las electroválvulas es la siguiente:

```

def abrirElectrovalvula(electrovalvula):
  if electrovalvula ==1:
    print ('electrovalvula 1\n')
    GPIO.output(electrovalve1, GPIO.LOW)
    GPIO.output(electrovalve2, GPIO.HIGH)
    GPIO.output(electrovalve3, GPIO.HIGH)
    GPIO.output(electrovalve4, GPIO.HIGH)
    GPIO.output(electrovalve5, GPIO.HIGH)
  elif electrovalvula == 2:
    print ('electrovalvula 2\n')
    GPIO.output(electrovalve1, GPIO.HIGH)
    GPIO.output(electrovalve2, GPIO.LOW)
    GPIO.output(electrovalve3, GPIO.HIGH)
    GPIO.output(electrovalve4, GPIO.HIGH)
    GPIO.output(electrovalve5, GPIO.HIGH)
    GPIO.output(electrovalve5, GPIO.HIGH)
  elif electrovalvula == 3:
    print ('electrovalvula 3\n')
    GPIO.output(electrovalve1, GPIO.HIGH)
```

```
    GPIO.output(electrovalve2, GPIO.HIGH)
    GPIO.output(electrovalve3, GPIO.LOW)
    GPIO.output(electrovalve4, GPIO.HIGH)
    GPIO.output(electrovalve5, GPIO.HIGH)
  elif electrovalvula == 4:
    print ('electrovalvula 4\n')
    GPIO.output(electrovalve1, GPIO.HIGH)
    GPIO.output(electrovalve2, GPIO.HIGH)
    GPIO.output(electrovalve3, GPIO.HIGH)
    GPIO.output(electrovalve4, GPIO.LOW)
    GPIO.output(electrovalve5, GPIO.HIGH)
  elif electrovalvula == 5:
    print ('electrovalvula 5\n')
    GPIO.output(electrovalve1, GPIO.HIGH)
    GPIO.output(electrovalve2, GPIO.HIGH)
    GPIO.output(electrovalve3, GPIO.HIGH)
    GPIO.output(electrovalve4, GPIO.HIGH)
    GPIO.output(electrovalve5, GPIO.LOW)
```

## 4.6. Control del circuito de succión

El circuito de succión está formado por el motor y el circuito de potencia explicado en la 3.7.1.4 diseñado para ser capaz de dar la potencia necesaria al motor. Para poder llevar a cabo el control del motor será necesario hacer uso de un pin de la BBB, en este caso se usa el P9_21 el cual es un pin que genera una señal PWM. La configuración de este pin se lleva acabo de la siguiente forma:

```
import Adafruit_BBIO.PWM as PWM
motorPin = 'P9_21' ##Motor de succion
```

La librería "Adafruit_BBIO.PWM" contiene las funciones del PWM, para controlar el motor y la temperatura variable de calentamiento del sensor. Por tanto, para poder controlar el motor se usan las siguientes funciones incluidas en las librerías de "Adafruit_BBIO".

1. Función para arrancar la succión motor la cual tiene 2 parámetros de entrada, por una parte el "motorPin" en donde se le indica el pin de la BBB el cual se va usar para configurarlo como salida PWM y por otra parte el parámetro "valor" el cual es ciclo PWM que se quiere dar a la señal, este valor puede estar entre el 0 % y el 100 %.

```
    PWM.start(motorPin, valor)

```

2. Función para detener la señal PWM que controla la succión del motor la cual recibe como parámetro de entrada el pin de la señal PWM que se quiere detener.

```
    PWM.stop(motorPin)

```

## 4.7. Circuitos de modulación

Después de explicar el funcionamiento de los circuitos de control de electroválvulas, motor y sensor de temperatura de forma individual se debe pasar a la unificación de estos circuitos llevando a cabo el funcionamiento completo de la placa de experimentación. Por tanto, el siguiente paso será detallar la configuración de los circuitos de modulación en amplitud y frecuencia. Por ende sera necesario dar las especificaciones de funcionamiento para cada una de las modulaciones tales como son los pines de conexión, configuración software de cada una de ellas, para así poder realizar pruebas aplicando cada una de las modulaciones desarrolladas en [5].

Para esta primera etapa lo primero que se hará es definir las muestras de odorantes que se van a usar y las concentraciones de estas mismas. Estas muestras de odorantes son disoluciones entre el odorante y el agua, esto se hace para evitar que el sensor se sature. En la tabla 4.3 se detallan las muestras que se han usado. La concentración del odorante se muestra en porcentaje y se interpreta de la siguiente manera:

- Muestra #0 de metanol con un 1,25 %, esto quiere decir que sobre una disolución de 10 ml, 125 ul son de metanol y los 9,875 ml restantes son de agua.

| | Metanol | Etanol | Butanol | Metanol |
|---|---|---|---|---|
| Muestra #0 | 1,25 % | 2,5 % | 1 % | 1,25 % |

Tabla 4.3: concentraciones muestras de odorantes.

En el desarrollo de este trabajo ha sido importante aprender como realizar las muestras que se van usar en los experimentos, en el anexo H se explica de forma detallada como se lleva a cabo la preparación de estas muestras. Tras la definición de las muestras con las cuales se va a trabajar, en la tablas 4.4 y 4.5 se definen los pines usados por el circuito de amplitud y los pines usados para el circuito de frecuencia así como la función de cada pin.

| **Puerto** | **Función** |
|---|---|
| P8_10 | Electroválvula 1 |
| P8_11 | Sensor temperatura y humedad |
| P8_12 | Electroválvula 2 |
| P8_14 | Electroválvula 3 |
| P8_16 | Electroválvula 4 |
| P8_18* | Electroválvula 5 |
| P9_21 | Pin motor succión |
| P9_22 | Pin calentamiento TGS2600 |
| P9_40 | Pin lectura ADC |

Tabla 4.4: Pines de conexión para el circuito de modulación en amplitud. (*) Pin disponible para la segunda placa de experimentación.

Como se han realizado dos placas de experimentación es importante resaltar que con la primera placa se han usado 4 muestras de odorante bajo estudio y con la segunda placa se han usado 5 muestras de odorante. En la tabla 4.6 se puede ver la asignación de odorantes y el numero de electroválvula por el cual sera succionado el odorante en cada una de las placas.

### 4.7.1. Adquisición de odorante sin modulación

La primera prueba que se va a realizar en la placa es la adquisición de odorante sin aplicar ningún tipo de modulación. En el proceso de adquisición de muestras de odorante el sensor

| Puerto | Funcion |
|---|---|
| P8_10 | Electrovalvula 1 |
| P8_11 | Sensor temperatura y humedad |
| P8_12 | Electrovalvula 2 |
| P8_14 | Electrovalvula 3 |
| P8_16 | Electrovalvula 4 |
| P8_18* | Electrovalvula 5 (Versión 2 de la placa) |
| P9_12 | Salida LM555 |
| P9_14 | Pin calentamiento TGS2600 |
| P9_21 | Pin motro succión |

Tabla 4.5: Pines de conexión para el circuito de modulación en frecuencia.

| | Numero de electroválvula | | | | |
|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** |
| Placa1 | Metanol | Etanol | Butanol | Aire | |
| Placa2 | Metanol | Etanol | Butanol | Aire | Metanol |

Tabla 4.6: Asignación de las electroválvulas y odorantes para cada una de las versiones de la placa diseñada.

trabajara aplicando una tensión fija de calentamiento, que en este caso es una señal PWM que suministra la tensión de calentamiento al sensor y que se vera reflejado como una temperatura en el calentador de este mismo.

Con esta primera prueba se ha hecho un primer acercamiento de cómo se comporta el sensor ante los distintos odorantes y es lo que ha dado pie a querer hacer un estudio de la respuesta obtenida por el sensor ante los distintos odorantes y ver cómo se comporta ante ellos. Este estudio se llevara a cabo en 4.8.

Para la adquisición sin modulación se realizaron pruebas con el sensor y cada uno de los odorantes, la idea que se tuvo es poder crear un vector de muestras conformado por el aire y alguno de los odorantes, de tal forma que el aire estuviese siendo succionado durante 2 minutos y el odorante bajo estudio durante 1 minuto esto de forma repetida durante cierto periodo de tiempo. Con esto lo que se consigue es la reproducibilidad del experimento y ver cual es el comportamiento del odorante.

El vector conmutación de electroválvulas que se usará dentro del programa para la adquisición de muestras sin modulación tiene la siguiente forma:

[4,4, valvula,4,4, valvula, 4,4, valvula, 4,4, valvula, 4,4, valvula,4,4]

En donde *valvula*, será un parámetro de entrada del programa principal y se corresponde con el valor de la electroválvula que dejara el paso del odorante durante el experimento. Para ver que válvula se corresponde con cada odorante ver tabla 4.6. En este caso, el valor 4 se corresponde a la electroválvula en la que se que succiona el aire. Por ejemplo, si el parámetro de entrada *valvula* fuese el valor 3 el vector de conmutación de electroválvulas quedara de la siguiente manera:

[4,4,3,4,4,3, 4,4,3,4,4,3,4,4,3,4,4]

En donde el valor 3 se corresponde a la electroválvula que deja el paso del butanol tanto en la primera como en la segunda versión de la plataforma y el valor 4 a la electroválvula que deja el paso del aire en también en las dos plataformas.

En este primer experimento lo que se ha querido hacer es ver la reproducibilidad de la lectura del sensor ante una entrada de tensión de calentamiento que sea constante. Para esta modulación se ha desarrollado un código basado en el código desarrollado en [5]. Este código permite la conmutación entre las electroválvulas de cada odorante durante cierto tiempo, mientras se realiza la lectura del puerto ADC de forma periódica (1 lectura por segundo). El código para la adquisición de odorante sin modulación usado para este primer experimento se puede encontrar en el anexo I.2.

#### 4.7.1.1. Pasos para lanzar un experimento adquisición de odorante sin modulación

Ahora se va a detallar el proceso para poder lanzar un experimento para la adquisición de odorante sin modulación, los pasos a seguir son los siguientes:

1. Antes de encender la BBB comprobar que esta la tarjeta microSD en la ranura correspondiente, ya que en esta unidad es donde se almacenarán los datos de los experimentos que se realicen.

2. Comprobar que el sensor TGS2600 esta conectado en la placa de experimentación en el conector correspondiente para esta modulación, en este caso sobre el conector que tiene la serigrafía "TGS_AMP" sobre la placa.

3. Revisar las conexiones que hay entre las placas del circuito de protección, placa de experimentación y la BBB. En la tabla 4.4 se encuentran especificados los pines de conexión desde la placa de experimentación hacia la BBB

4. Revisar que el cable de Ethernet este conectado a la BBB.

5. Conectar la fuente de alimentación de 5V a la BBB, y esperar hasta que esta este encendida y se puede acceder al sistema. Una vez dentro situarse sobre el directorio /home/debian que es donde se encuentra el código para lanzar el experimento.

6. A continuación conectar la fuente externa de 24V a la placa de experimentación.

7. Una vez estén las dos placas encendidas, lo primero que se hará es apagar el pin de la BBB que controla la succión del motor ya que este pin esta encendido por defecto y por tanto el motor estará funcionado. Para apagar el pin correspondiente sera necesario ejecutar el siguiente comando:

   python3 inicio.py

8. Una vez apagado el motor, se debe ejecutar el siguiente comando para llevar a cabo el experimento:

   python3 puro.py succión heat2600 tiempo cambio válvula

   En donde succión, heat2600, tiempo,cambio y válvula son parámetros que el usuario debe introducir. El valor de la succión debe ser un valor entre 65 % y 100 %, y esto representa la potencia de succión del motor. Heat2600 es la tensión de calentamiento que se va aplicar al sensor durante todo el experimento, esta tensión puede recibir un valor de entre 0 % y el 100 % que se corresponde al ciclo PWM. El parámetro tiempo es un entero y sera la duración en minutos del experimento. Cambio es un entero y es el tiempo de conmutación entre electroválvulas en segundos.

9. Una vez terminado el experimento se recomienda apagar la plataforma de experimentación si esta no va hacer usada. Para apagar la BBB se lanzara el siguiente comando:

   shutdown -h now

   Se debe esperar hasta que la placa se apague, esto se comprobara por que los LEDs de la BBB se apagarán.

10. Una vez apagada la BBB es conveniente desconectar primero la fuente de alimentación de 24V de la placa de experimentación y a continuación la fuente de alimentación de 5V de la BBB.

#### 4.7.1.2. Ejemplo de experimento de adquisición de odorante sin modulación

Para llevar a cabo este experimento se deben realizar los pasos indicados en 4.7.1.1 hasta el punto 8, aquí se lanzara el siguiente comando:

python3 puro.py 70 100 17 60 1

En este caso se lanza un experimento de una duración de 17 minutos y conmutación de electroválvulas cada 60 segundos. La succión del motor es de un 70 % y la tensión de calentamiento del sensor sera del 100 %. En este caso el experimento se ha realizado para el metanol que se corresponde con al electroválvula 1 de la primera versión de la placa de experimentación. Para los demás odorantes se usan los mismos parámetros para lanzar el experimento excepto que se cambia el valor de la válvula (1- Metanol, 2- Etanol, 3- Butanol, 5 - Odorante 1 para la segunda placa de experimentación ) que se quiere abrir para el odorante bajo estudio.

En la figura 4.6, se puede observar el comportamiento de los distintos odorantes cuando al sensor se le aplica una tensión de calentamiento del 100 % del ciclo PWM que se correspondería con 5 V. La lectura del sensor para el metanol llegar alcanzar los 1000 mV, para el etanol se obtienen valores de lectura entorno a los 1300 mV y para el butanol se obtienen valores de lectura cercanos a los 1000 mV. Para las transiciones del aire se observa como el valor de lectura es cercano a los 200 mV.

En la figura 4.7, se observan los resultados del experimento realizado sobre la segunda placa de experimentación. En esta se observa por una parte como los resultados obtenidos no son similares a los obtenidos en la figura 4.6, esto puedo ser causado por que la concentración de las muestras hayan variado mínimamente. Además, es importante tener en cuenta que el motor usado en la segunda placa no es el mismo que se usa en la primera placa y esto puede causar que al momento de succionar el odorante la cantidad de odorante que el sensor capte no sea la misma debido las características del motor. Por otra parte, en la figura 4.7 en el panel superior se representa la adquisición de metanol que proviene de la primera y la quinta electroválvula y que están representadas con trazo de color azul y rojo respectivamente. En este caso, se observa como las dos señales coinciden viniendo cada una de una electroválvula diferente en este caso el metanol ubicado en quinta electroválvula es el odorante que mayor camino debe recorrer durante el proceso de succión hasta llegar al sensor.

Por tanto, esto puede hacer intuir que un mismo odorante, uno ubicado a una distancia moderada del sensor y el otro ubicado también a otra distancia moderada del sensor pueden presentar un mismo patrón estando bajo las mismas condiciones. Por lo que se podría asumir que a distancias moderadas del sensor la respuesta de un mismo odorante ante el sensor no debería cambiar de forma drástica.

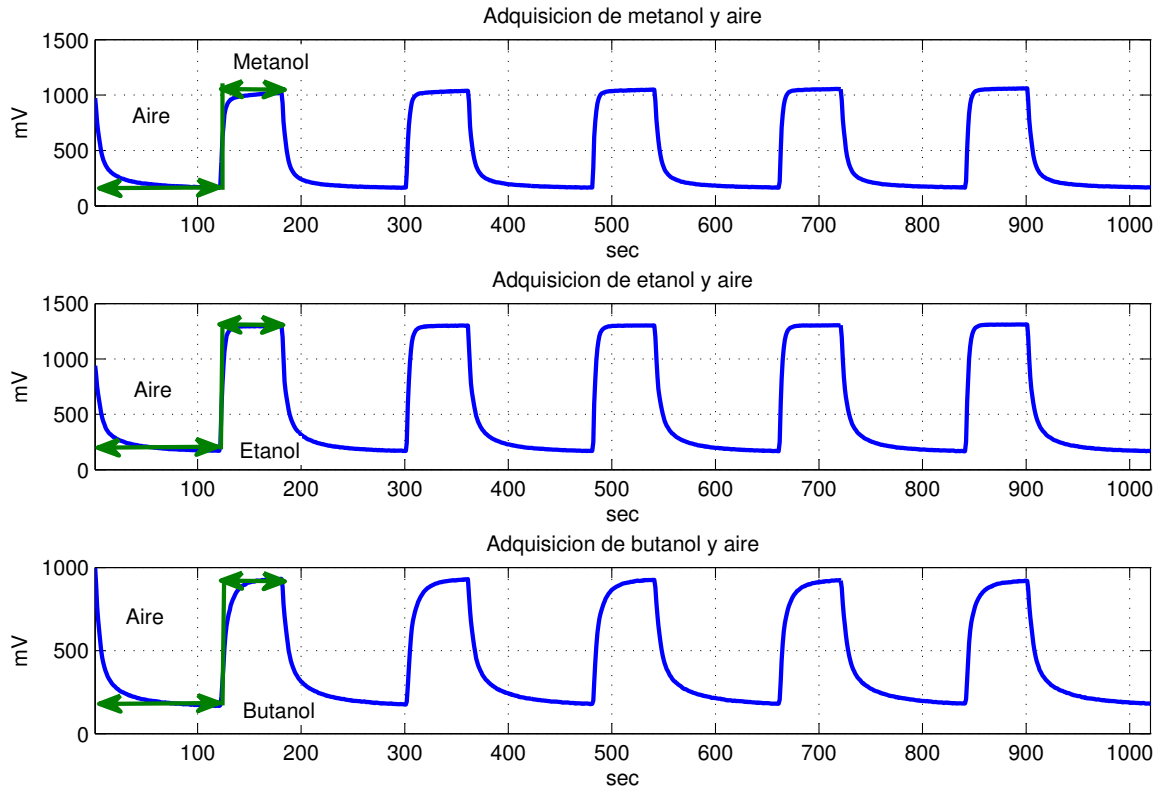



Figura 4.6: Adquisición de aire durante 2 minutos y los distintos odorantes durante 1 minuto en un periodo de 17 minutos de experimentación, en la primera placa de experimentación.

Observando los resultados de las figuras 4.6 y 4.7, como se ha comentado anteriormente los resultados obtenidos no son similares y esto hace pensar que hay algo mal dentro del sistema. Como se ha dicho algunos motivos podrían ser el cambio de concentración de las muestras ya que estas se van evaporando a medida que se hacen experimentos y pueden cambiar en esto caso las muestras antes de realizar cada uno de los experimentos fueron cambiadas para que así los dos experimentos partiesen en las mismas condiciones. Por otra parte, también se plantea el hecho de que al usar motores distintos en cada una de las placas puede que hayan pequeñas variaciones esto no debería ser algo que cambiase de forma drástica la respuesta del sensor ya que en el habitáculo del sensor siempre habrá una misma concentración de odorante. Por tanto, se plantea el hecho de que el en el sistema de electroválvulas haya alguna fuga debido a que no estén correctamente conectadas las electroválvulas con la base de montaje. Para ello primero que se hace es realizar un adquisición solo del aire en cada una de las electroválvulas para ver cual es el comportamiento.

En la figura 4.8, en el panel de la derecha se observa los resultados obtenidos de la lectura del aire en cada una de las electroválvulas. Como se puede observar la lectura de las electroválvulas 1,2 y 4 presenta algunas oscilaciones que pueden ser causadas por que no estén bien conectadas las electroválvulas. Por tanto, se decide desmontar las electroválvulas de la base y limpiar esta para así evitar que haya polvo sobre la superficie de la base que pueda producir la obstrucción del paso del odorante.

Después de desmontar las electroválvulas de las base de montaje y volverlas a poner, se realiza de nuevo el mismo experimento para ver si ha habido algún cambio. En el panel de la derecha del figura 4.8, se observan los resultados obtenidos de la lectura del aire en cada una

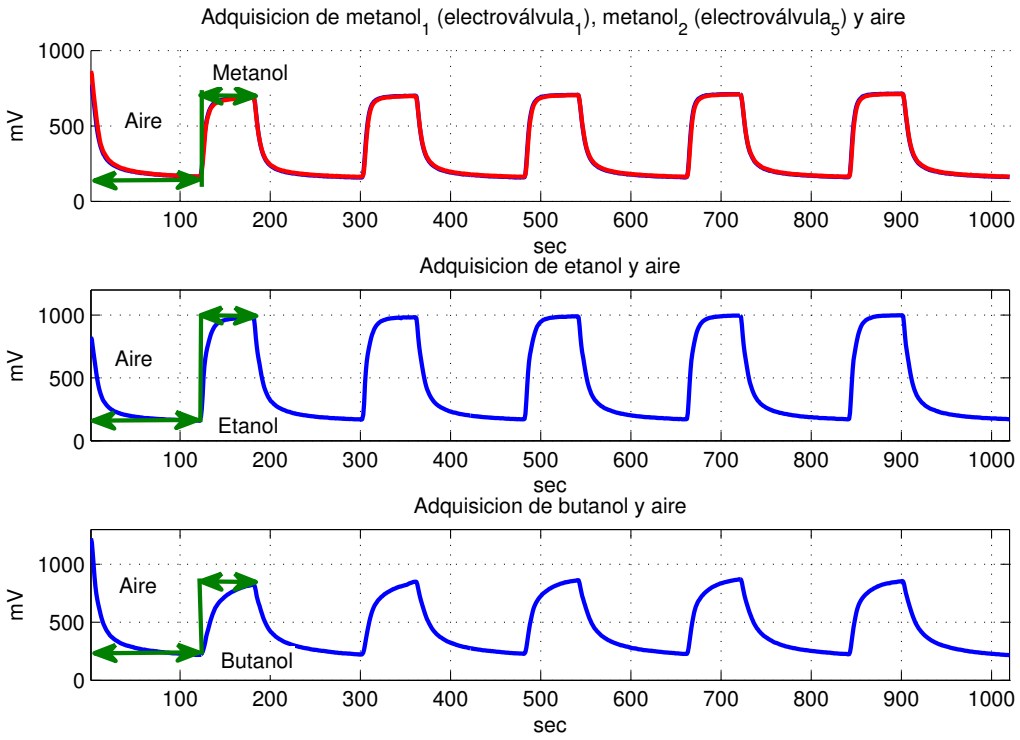



Figura 4.7: Adquisición de aire durante 2 minutos y los distintos odorantes durante 1 minuto en un periodo de 17 minutos de experimentación, en la segunda placa de experimentación. En el panel superior se muestra lectura del metanol ubicado en la primera electroválvula (linea azul) y superpuesto esta el metanol ubicado en la quinta electroválvula (línea roja).

de las electroválvulas. En este caso se ve como ya no hay oscilaciones en ninguna de las lectura correspondiente a cada electroválvula. También es importante tener en cuenta que se esperaría que todas las lecturas del aire en cada electroválvula tendiese a ser la misma.

Tras comprobar que el montaje de las electroválvulas estaba mal conectado debido a que quizás no se estuviese ejerciendo la presión necesaria o se haya movida mínimamente alguna electroválvula de su posición de conexión con respecto a la base, se decide volver a realizar el experimento de modulación sin modulación para la segunda placa de experimentación con los mismos parámetros de los experimentos anteriores.

En la figura 4.9, se observan los resultados obtenidos para la adquisición de odorante sin modulación sobre la segunda placa de experimentación. Como se observa los resultados obtenidos son muy similares a lo que se se han obtenido en la figura 4.6, además se observa en el panel superior como la lectura del metanol en la primera electroválvula coincide con la lectura del metanol ubicado en la quinta electroválvula.

Para los distintos experimentos realizados en cada una de las placas, se ve que hay un patrón común al principio de cada uno de ellos y es que en las primeras lecturas del sensor se puede observar como hay un pequeño pico de tensión que disminuye inmediatamente. Esto se debe a que en las lecturas de las primeras muestras puede que hayan quedado impurezas de odorante en el habitáculo del sensor. Es importante también resaltar el hecho de que en las figuras 4.6, 4.7 y 4.9 se puede apreciar la huella que deja cada uno de los odorantes cuando se lleva a cabo una transición entre el odorante bajo estudio y el aire. Además con estos experimentos se ha podido ver diferencias entre los resultados obtenidos que han sido útiles para encontrar un problema del sistema que hace que haya cambios notorios en la respuesta obtenida por el sensor y que

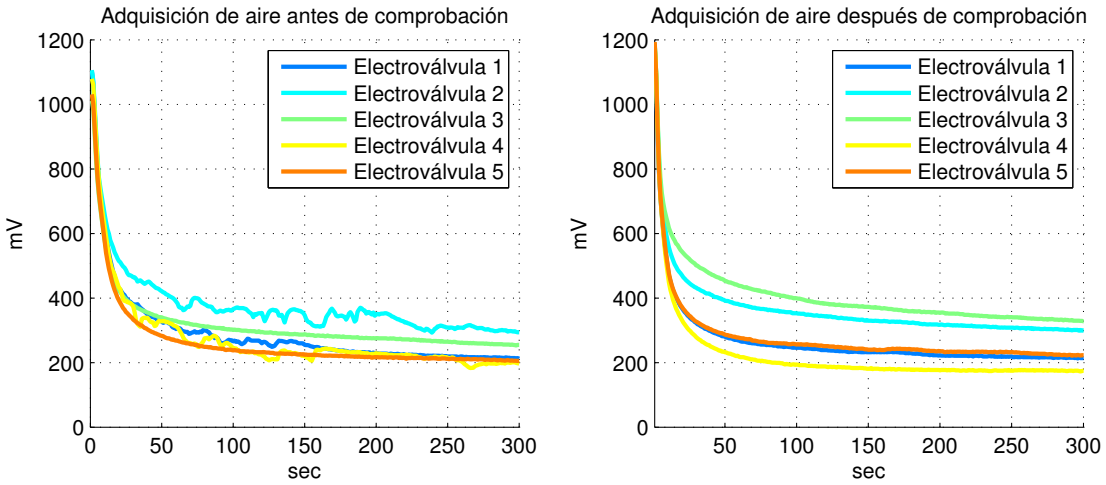



Figura 4.8: Respuesta del sensor ante el aire en cada una de las electroválvulas. En el panel de la derecha se muestra la respuesta antes de realizar la comprobación del sistema de electroválvulas y en panel izquierdo se muestra la respuesta después de la comprobación.

por ende ha servido para investigar lo que estaba pasando y dar una solución al problema. Por ultimo, es importante tener en cuenta como en el momento en que se succiona el aire se observa un proceso de limpieza del odorante al cual estaba expuesto el sensor.

### 4.7.2. Modulación de regresión de temperatura

La modulación por regresión de temperatura es una de las modulaciones que fue implementada en [5]. En esta modulación, la temperatura de calentamiento es autoadaptada basándose en el valor obtenido previamente. Esto quiere decir que para este tipo de modulación se usa un circuito en lazo cerrado en el que la salida del sensor es un parámetro que se usa para calcular la siguiente temperatura que se debe aplicar. De esta forma se consigue que señal del sensor varíe dinámicamente al entrar en interacción con el gas, consiguiendo patrones característicos para cada odorante [1].

El calculo de la nueva temperatura se lleva a cabo a través de un calculo por regresión lineal basado en la tendencia de las medidas previas del odorante bajo estudio, conocido como $SLOPE$, además de aplicar un parámetro fijo denominado $TENDENCIA$ el cual sirve para determinar el cambio de amplitud de la temperatura. En la ecuación 4.1 se define el calculo de la nueva temperatura.

$$T_{heat\_new} = T_{heat\_old} - (SLOPE \cdot TENDENCIA). \tag{4.1}$$

Por tanto, se debe comprobar sobre las placas construidas que esta modulación funciona correctamente. En este caso, los códigos usados para esta modulación han sido desarrollado en [33], que permite que la realización de los experimentos sea mas versátil.

#### 4.7.2.1. Pasos para lanzar un experimento de la modulación de regresión en temperatura

Ahora se va a detallar el proceso para poder lanzar un experimento con una modulación de regresión en temperatura, los pasos a seguir son los siguientes:

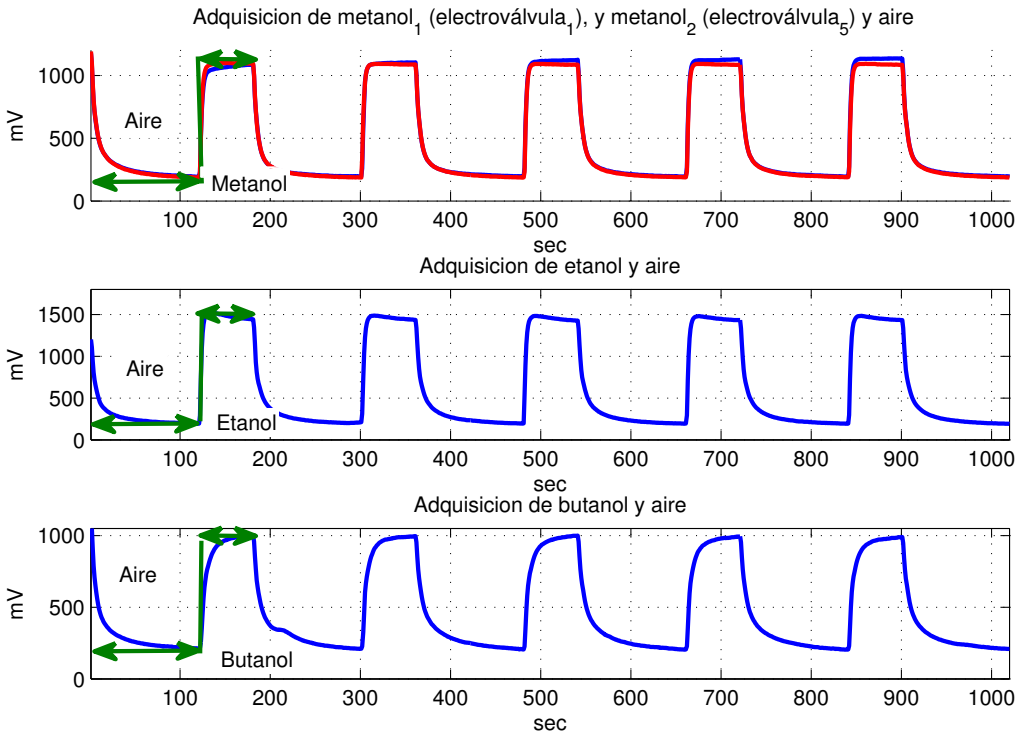



Figura 4.9: Adquisición de aire durante 2 minutos y los distintos odorantes durante 1 minuto en un periodo de 17 minutos de experimentación, en la segunda placa de experimentación después de la revisión del sistema de electroválvulas. En el panel superior se muestra lectura del metanol ubicado en la primera electroválvula (linea azul) y superpuesto esta el metanol ubicado en la quinta electroválvula (línea roja).

1. Comprobar que el sensor TGS2600 esta conectado en la placa de experimentación en el conector correspondiente para esta modulación, en este caso sobre el conector que tiene la serigrafía "TGS_AMP" sobre la placa.

2. Revisar las conexiones que hay entre las placas del circuito de protección, placa de experimentación y la BBB. En la tabla 3.5 se encuentran los pines que se deben comprobar que esta conectados correctamente.

3. Revisar que el cable de Ethernet este conectado a la BBB.

4. Conectar la fuente de alimentación de 5 V a la BBB, y esperar hasta que esta este encendida y se puede acceder. Una vez dentro situarse sobre el directorio /home/debian/codigosNuevos que es donde se encuentra el código para lanzar el experimento.

5. A continuación conectar la fuente externa de 24 V a la placa de experimentación.

6. Una vez estén las dos placas encendidas, lo primero que se hará es apagar el pin de la BBB que controla la succión del motor ya que este pin esta encendido por defecto y por tanto el motor estará funcionado. Para apagar el pin correspondiente sera necesario ejecutar el siguiente comando:

   python3 inicio.py

7. Ahora se ejecutara el siguiente comando para llevar a cabo el experimento para la modulación en regresión de temperatura:

```
python3 PyHuele.py script_regresion.txt configuracion_plataforma.txt
```

En donde "PyHuele.py" es el ejecutable para la realización del experimento y los archivos "script_regresion".txt y "configuracion_plataforma.txt", son ficheros de configuración de la modulación que se quiere realizar y que se detallan en la sección 4.7.2.2.

8. Una vez terminado el experimento se recomienda apagar la plataforma de experimentación si esta no va hacer usada. Para apagar la BBB se lanzara el siguiente comando:

```
shutdown -h now
```

Se debe esperar hasta que la placa se apague, esto se comprobara por que los LEDs de la BBB se apagarán.

9. Una vez apagada la BBB es conveniente desconectar primero la fuente de alimentación de 24 V de la placa de experimentación y a continuación la fuente de alimentación de 5 V de la BBB.

#### 4.7.2.2. Ejemplo de experimento modulación de regresión de temperatura

Para la realización del experimento de deberá seguir los pasos descritos en la sección 4.7.2.1 hasta le punto 8. Antes de lanzar el comando para la realización del experimento se va a explicar el contenido de los dos ficheros de configuración que se deben usar.

Por una parte tenemos el fichero de configuración que se llama "script_regresion.txt" este fichero contiene la información acerca de el experimento que se va realizar. A continuación, se muestra el contenido de este fichero:

```
1   number_experiments: 1
    suction: 70*
3   duration_stimulation: 120*
    initial_samples: 60*
5   number_samples: //
    time_between_stimulus: 0*
7   name_file: regresion*
    name_folder: Regresion*
9   experiment_version: 2
    modulation: 2
11  tendency: 5*
    heat_sensor: 50*
13
```

Los parámetros que se configuran en este fichero son los siguientes:

- number_experiments, con este parámetro se define el numero de experimentos que se quieren realizar.
- suction, representa la potencia que tendrá el motor.
- duration_stimulation, representa el tiempo en segundos en el que se succionara un determinando odorante.
- initial_samples, son las muestras iniciales que se leen del sensor.
- number_samples, representa el numero de muestras que aleatorias que contendrá el experimento.

- time_between_stimulus, este es el tiempo que hay entre una muestra y otra dentro del experimento. Entre una muestra y otro lo que se hace es succionar aire. Si el valor de este parámetro es 0, no hay succión de aire en la transición de lectura de una muestra a otra.
- name_file, es el nombre que recibe el archivo de texto donde se almacena los datos.
- name_folder, es el nombre que recibirá la carpeta donde se almacenan los ficheros generados durante el experimento.
- experiment_version, con este parámetro se indica a la plataforma que odorantes son los que se quieren analizar. Los valores posibles son los siguientes:
  1. Con este valor la plataforma analizará todos los odorantes, por tanto se consigue que se analicen todas las transiciones entre los odorantes.
  2. Con este valor la plataforma escoge los odorantes que se van analizar de forma aleatoria.
  3. Con este valor el usuario debe introducir odorantes que se quieren analizar.
- modulation, especifica el tipo de modulación que se va a realizar en el experimento, en este caso la modulación es la de regresión de temperatura y se corresponde con el código 2.
- tendency, es el parámetro de tendencia de cuanto quiero que suba o baje la temperatura en la modulación de regresión.
- heat_sensor, es la temperatura que se va aplicar al sensor este valor puede estar entre el 0 % y el 100 %.
- plot_mode, este valor sirve para la representación de lo datos. En este proyecto esa opción esta deshabilitada.

Algunos de los parámetros del fichero "script_regresion.txt" contienen el símbolo *, esto quiere decir que en el caso de que el numero de experimentos fuese mas de 1, los otros experimentos tendrán en estos mismos parámetros de configuración.

El siguiente fichero es "configuracion_plataforma.txt", el cual contiene configuración de la placa este fichero es común para todas las modulaciones que soporta el lenguaje de experimentación desarrollado en [33]. A continuación, se muestra el contenido de este fichero para la configuración de la primera placa y luego el contenido para la segunda placa:

```
#Contenido fichero de configuracion para la primera placa
 reading_port: P9_40
 motor_pin: P9_21
 resistance: 470
 SD_folder: CAPTURAS/REGRESION
 valve_position: R
 heating_port: P9_22
 sleep_time: 1
 electrovalves_port: P8_10,P8_12,P8_14,P8_16

```

```
#Contenido fichero de configuracion para la segunda placa
 reading_port: P9_40
 motor_pin: P9_21
 resistance: 470
 SD_folder: CAPTURAS/REGRESION
 valve_position: R
```

```
7     heating_port: P9_22
      sleep_time: 1
9     electrovalves_port: P8_10,P8_12,P8_14,P8_16,P8_18
```

Los parámetros que se configuran en este fichero son los siguientes:

- reading_port, se configura el puerto de lectura del sensor TGS2600.
- motor_pin, se especifica el pin que controla la succion del motor.
- resistance, es el valor de la resistencia de carga $R_L$ que se usa a la salida del sensor.
- SD_folder, nombre de la carpeta principal que contiene todos los experimentos
- valve_position, define la posición en la que esta activa al electroválvula, en el caso de la placa la posición es R, normalmente cerrada.
- heating_port, determina el puerto por el cual se calienta el sensor TGS260
- sleep_time: tiempo de muestreo
- electrovalves_port: se configuran los puertos en los cuales se encuentran las electroválvulas.

Una vez configurados los dos ficheros previamente comentados, se puede lanzar el experimento para la modulación de regresión en temperatura. En la figura 4.10, se pude observar el resultado obtenida al realizar la modulación de regresión. La conmutación de electroválvulas es la siguiente [[4], [1], [1], [2], [1], [2]], siendo 1-Metanol, 2-Etanol, 3-Butanol y 4-Aire.

Las primeras 60 muestras se corresponden a las muestras iniciales que lee el sensor. La primera muestra del odorante entra en el segundo 61 y se observa como hay una bajada de tensión y por tanto la temperatura de calentamiento se aumenta con el fin de compensar esta caída de tensión. Cuando se produce una subida de tensión lo que se produce es el efecto inverso y la temperatura de calentamiento del sensor disminuye, la idea de esta modulación es poder conseguir estabilizar la señal.

Con este primer experimento, se comprueba como la modulación en amplitud funciona correctamente sobre la placa. Por tanto, el siguiente paso es comprobar la reproducibilidad del experimento. Para ello, se llevaron acabo experimentos en distintas fechas para comparar los resultados. Por tanto, partiendo del experimento anterior se vuelve a realizar el mismo experimento para comprobar cual es el resultado obtenido. Para poder lanzar el experimento de nuevo sera necesario configurar el fichero de configuración "script_regresion.tx", ya que esta vez no se quiere que el programa genere un vector aleatorio de las electroválvulas, si no que este vector se pasará por parámetro.

A continuación, se muestra el contenido del fichero "script_regresion.tx", en el que se agrega el parámetro "number_samples" se modifica ya que no se define el numero de muestras que se quieren registrar, también se debe modificar el parámetro "experiment_version" con un valor a 3 que indica que el usuario es quien introduce los odorantes que se van a analizar. Por último, se agrega el parámetro "vector_open_valves", el cual contiene los odorantes que introduce el usuario y que se analizan durante el experimento, en [33] se puede consultar mas información de como configurar ese último parámetro.

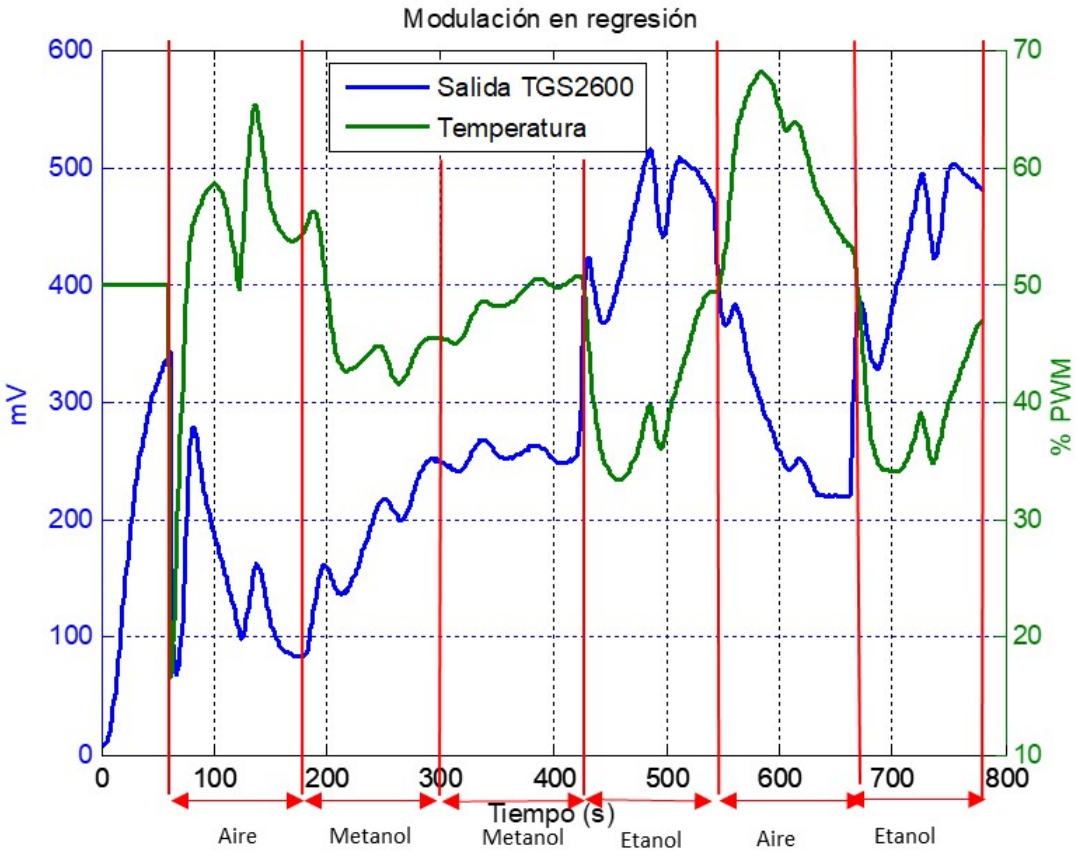



Figura 4.10: Prueba de la modulación en regresión sobre la primera placa de experimentación, con tensión inicial de calentamiento del sensor 50 % del ciclo PWM, succión del motor 70 % del ciclo PWM, con tendencia de medida igual a 5, 60 muestras iniciales y vector aleatorio de conmutación de electroválvulas.

```
number_experiments: 1
suction: 70*
duration_stimulation: 120*
initial_samples: 60*
number_samples: //
time_between_stimulus: 0*
name_file: regresion*
name_folder: Regresion*
experiment_version: 3
modulation: 2
tendency: 5*
heat_sensor: 50*
vector_open_valves: 4-1-1-2-1-2
```

Una vez modificado el fichero de configuración se puede lanzar el experimento de regresión. En la figura 4.11, se observan los resultados obtenidos. Al igual que en el experimento anterior tenemos que las primeras 60 muestras se corresponden con las muestras iniciales que se leen del sensor. Por tanto, al primera muestra del odorante entra en el segundo 61 y se observa como hay una caída de tensión a lo cual el sistema reacciona aumentando la temperatura de calentamiento.

Comparando los resultados que se muestran en la figura 4.10 con los de la figura 4.11, se

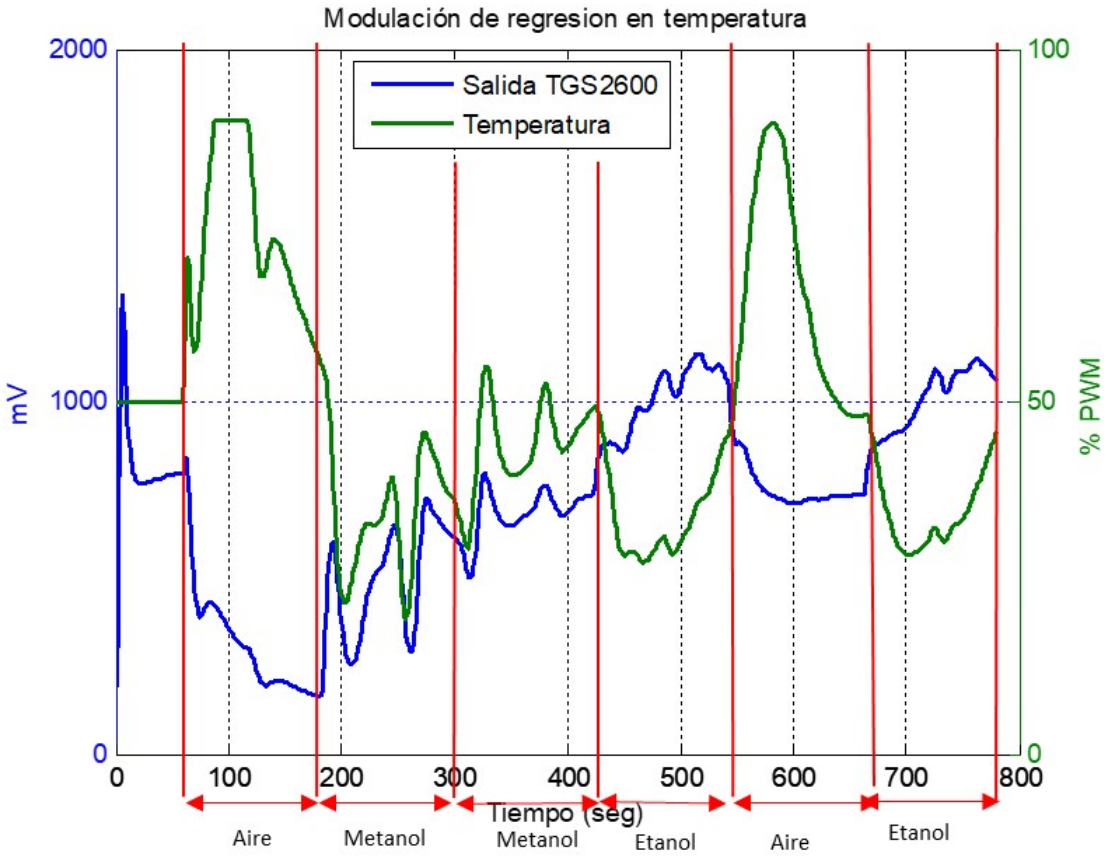



Figura 4.11: Prueba de la modulación en regresión sobre la placa de experimentación, con la misma transición entre odorantes como en 4.10.

puede observar que el patrón de las señales es muy similar excepto por los valores de lectura del sensor que son mas altos que en el primer experimento. En este caso podemos ver como no hay reproducibilidad ya que los experimentos están hechos en distintos periodos de tiempo y las condiciones iniciales no son las mismas en ambos casos. El hecho de que en el segundo experimento las lecturas del sensor obtengan valores mas altos que las del primer experimento puede ser dado por el cambio de las muestras de odorantes, ya que aunque las concentraciones son las mismas con el paso del tiempo estas concentraciones pueden ir variando.

Una vez comprobado el experimento anterior en el que no se ha producido reproducibilidad del mismo, se pasa a realizar un experimento en el cual se usen todos los odorantes con sus posibles transiciones. En este caso, se llevan a cabo varios experimentos en el que se pretende comprobar la reproducibilidad del experimento partiendo de las mismas condiciones iniciales excepto por el hecho de que no se cambian los odorantes y en el que se vera como evoluciona en el tiempo.

Lo primero que se debe hacer para este nuevo experimento es configurar el fichero correspondiente a la modulación "script_regresion.tx" como se muestra a continuación:

```
number_experiments: 40
suction: 70*
duration_stimulation: 120*
initial_samples: 60*
number_samples: //
time_between_stimulus: 0*
name_file: regresion*
name_folder: Regresion*
```

```
9  experiment_version: 1(40)
   modulation: 2
11 tendency: 5(40)
   heat_sensor: 50(40)
```

En el fichero se especifica que el numero de experimentos que se realizaran de forma seguida son 40 y todos ellos con los mismo parámetros de configuración que el primer experimento que se realizo.

En este caso, como se ha mencionado anteriormente el experimento contendrá todas las posibles transiciones entre los odorantes esto es el vector de electroválvulas de conmutación es [[1], [1], [2], [2], [3], [3], [4], [4], [2], [1], [3], [1], [4], [3], [2], [4], [1]], siendo 1-Metanol, 2-Etanol, 3-Butanol y 4-Aire.

En la figura 4.12 y 4.13, se representan los resultados de un experimento en el cual se han obtenido 40 capturas. Cada panel de las dos figuras representan las transiciones entre cada uno de los distintos odorantes. Por una parte, en las filas se representan el odorante desde donde se esta realizando actualmente la lectura del sensor y en las columnas se representan el odorante sobre el cual se hace la transición o cambio de odorante.

La figura 4.12, representa la lectura del sensor de cada uno de los cambios de estado entre odorantes. Como se puede observar los resultados muestran que hay una superposición entre los experimentos pero al mismo tiempo hay una pequeña desviación entre unos y otros. Esto puede ser debido por que la concentración del odorante disminuye a medida que se hace el proceso de succión.

Por otra parte, la figura 4.13 representa la temperatura que se aplica al sensor dependiendo del valor leído por este. Como se puede observar la temperatura aplicada al sensor en cada uno de los distintos cambios de estado es muy similar y hay poca diferencia entre el cambio de temperatura.

Por tanto, en este último experimento se comprueba la reproducibilidad del experimento ya que al observar las figuras 4.12 y 4.13, se observa como al superponer las 40 experimentos realizados estos son similares y la diferencia que hay se debe al gasto del odorante durante el tiempo, que como se indica en la figura los colores más fríos representan los experimentos mas antiguos. Por tanto, los experimentos con colores más cálidos son en los que las muestras de odorante han ido cambiando de concentración a medida que se llevaba a cabo cada experimento en el tiempo. Para terminar de comprobar esta técnica de modulación, solo quedaría por probar la técnica en la segunda placa construida. En este caso, se ha decidido comprobar el funcionamiento de esta técnica usando todas la transiciones posibles entre lo odorantes, que en este caso se tienen 5 odorantes. El vector de transiciones es el siguiente:[[1], [1], [2], [2], [3], [3], [4], [4], [5], [5], [4], [3], [2], [1], [3], [1], [4], [2], [4], [1], [5], [2], [5], [3], [5], [1]],siendo 1-Metanol, 2-Etanol, 3-Butanol, 4-Aire y 5-Metanol.

En las figuras 4.14 y 4.15, se muestran los resultados obtenidos para la segunda placa de experimentación para 3 experimentos que se han realizado en los que se representa la lectura obtenida por el sensor y la tensión de calentamiento aplicada al sensor. Como se puede observar en este caso aunque el numero de experimentos es menor al superponer los valores obtenidos en cada experimento se observa como las respuestas en cada uno de ellos es similar.

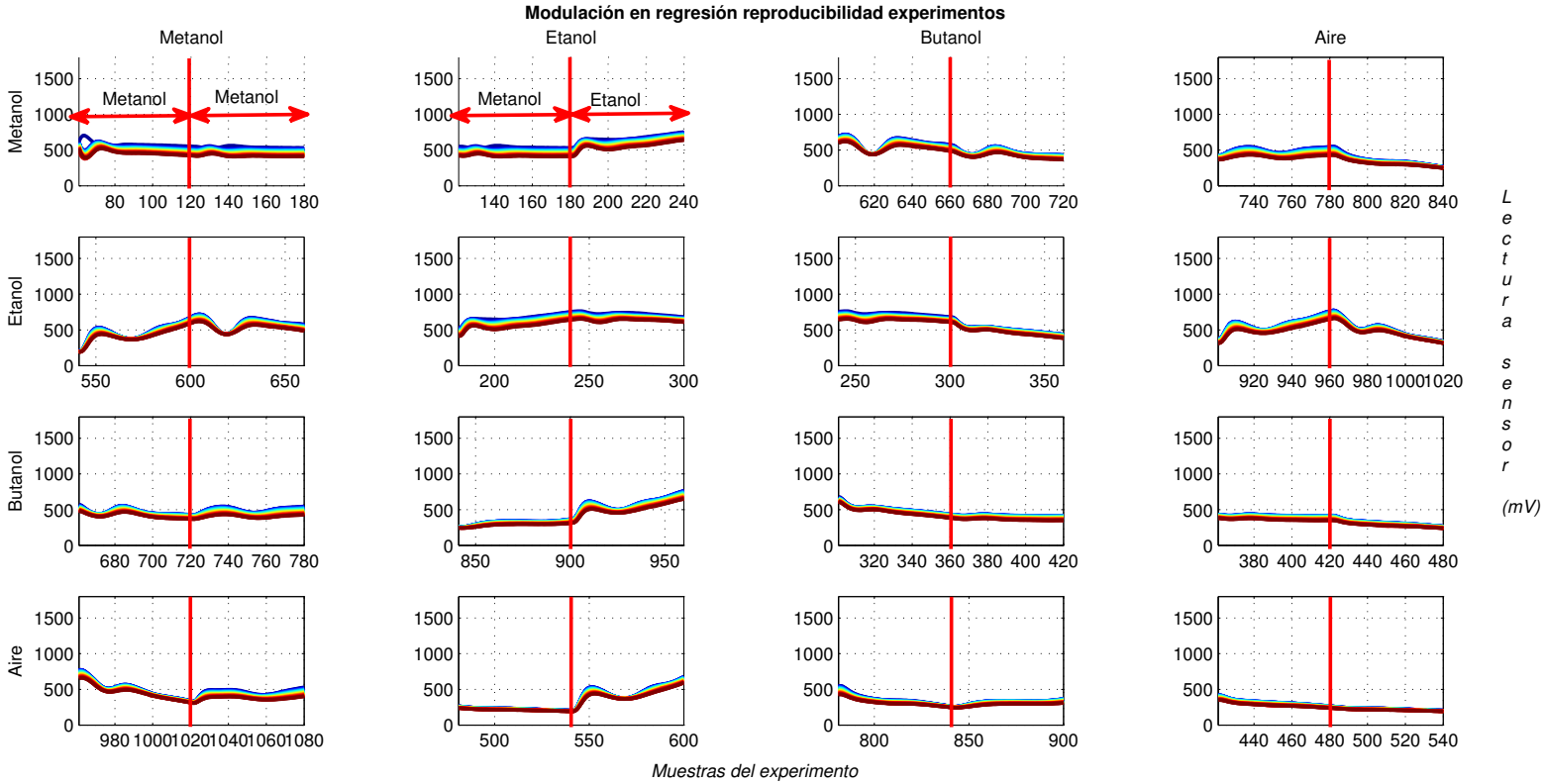



Figura 4.12: Experimento de reproducibilidad en la primera placa de experimentación de la modulación en amplitud para 40 experimentos,en el que se representa la lectura del sensor. Los colores fríos representan los experimentos más antiguos que los colores cálidos. En la figura se representa en la fila el odorante en donde se está realizando actualmente la lectura y en la columna el odorante al que se va hacer la transición (cambio de odorante). Cada uno de los paneles se lee de la siguiente forma: por ejemplo, metanol(fila)-metanol(columna) quiere decir que hasta la línea roja es metanol y luego se hace la transición al propio metanol. Otro ejemplo es metanol-etanol quiere decir que hasta la línea roja es metanol y luego se hace la transición al etanol.

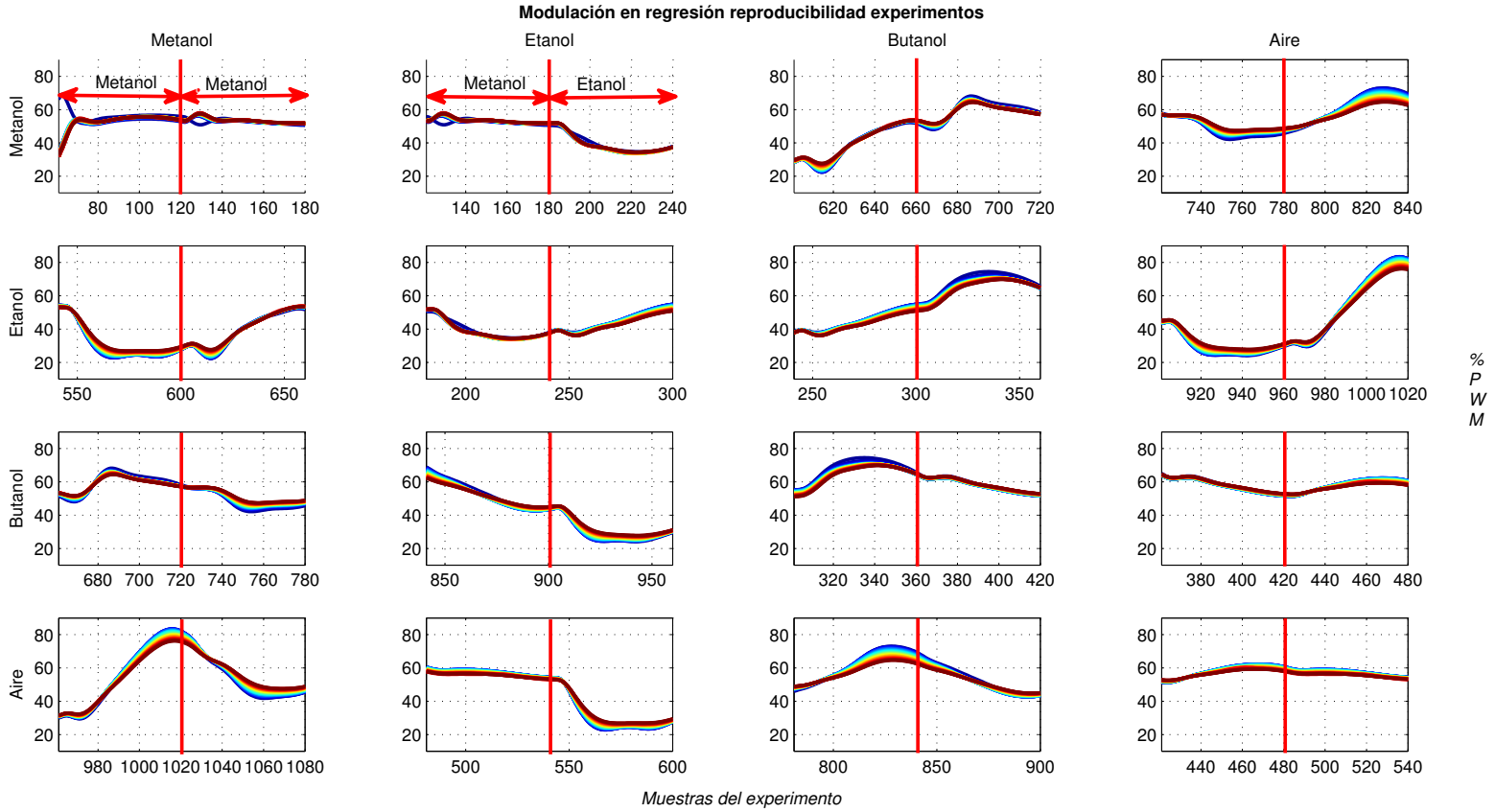



Figura 4.13: Experimento de reproducibilidad en la primera placa de experimentación de la modulación en amplitud para 40 experimentos, en el que se representa temperatura aplicada al sensor. Los colores fríos representan los experimentos más antiguos que los colores cálidos. En la figura se representa en la fila el odorante en donde se está realizando actualmente la lectura y en la columna el odorante al que se va hacer la transición (cambio de odorante). Cada uno de los paneles se lee de la siguiente forma: por ejemplo, metanol(fila)-metanol(columna) quiere decir que hasta la línea roja es metanol y luego se hace la transición al propio metanol. Otro ejemplo es metanol-etanol quiere decir que hasta la línea roja es metanol y luego se hace la transición al etanol.

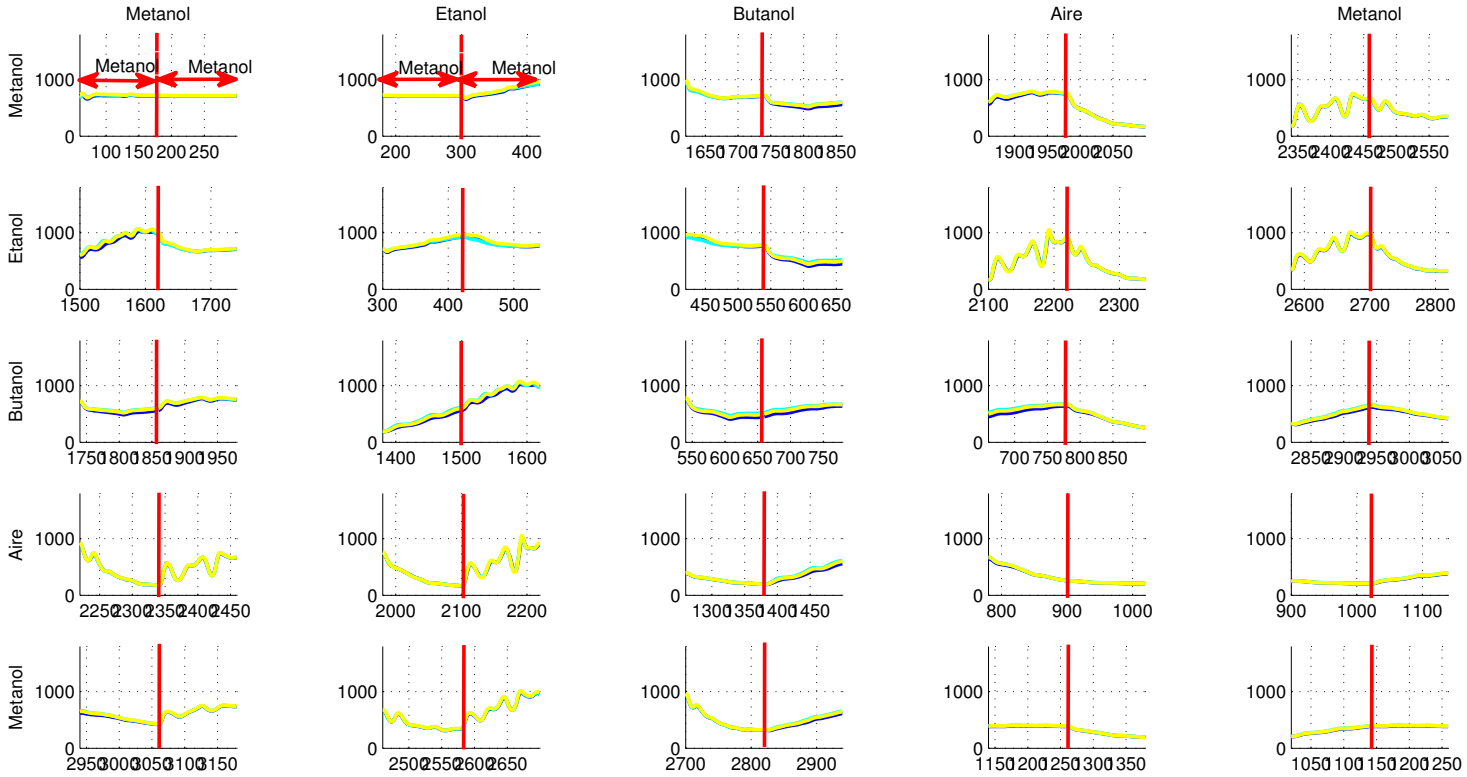



Figura 4.14: Experimento de reproducibilidad en la primera placa de experimentación de la modulación en amplitud para 3 experimentos,en el que se representa la lectura del sensor. En la figura se representa en la fila el odorante en donde se está realizando actualmente la lectura y en la columna el odorante al que se va hacer la transición (cambio de odorante). Cada uno de los paneles se lee de la siguiente forma: por ejemplo, metanol(fila)-metanol(columna) quiere decir que hasta la línea roja es metanol y luego se hace la transición al propio metanol. Otro ejemplo es metanol-etanol quiere decir que hasta la línea roja es metanol y luego se hace la transición al etanol.

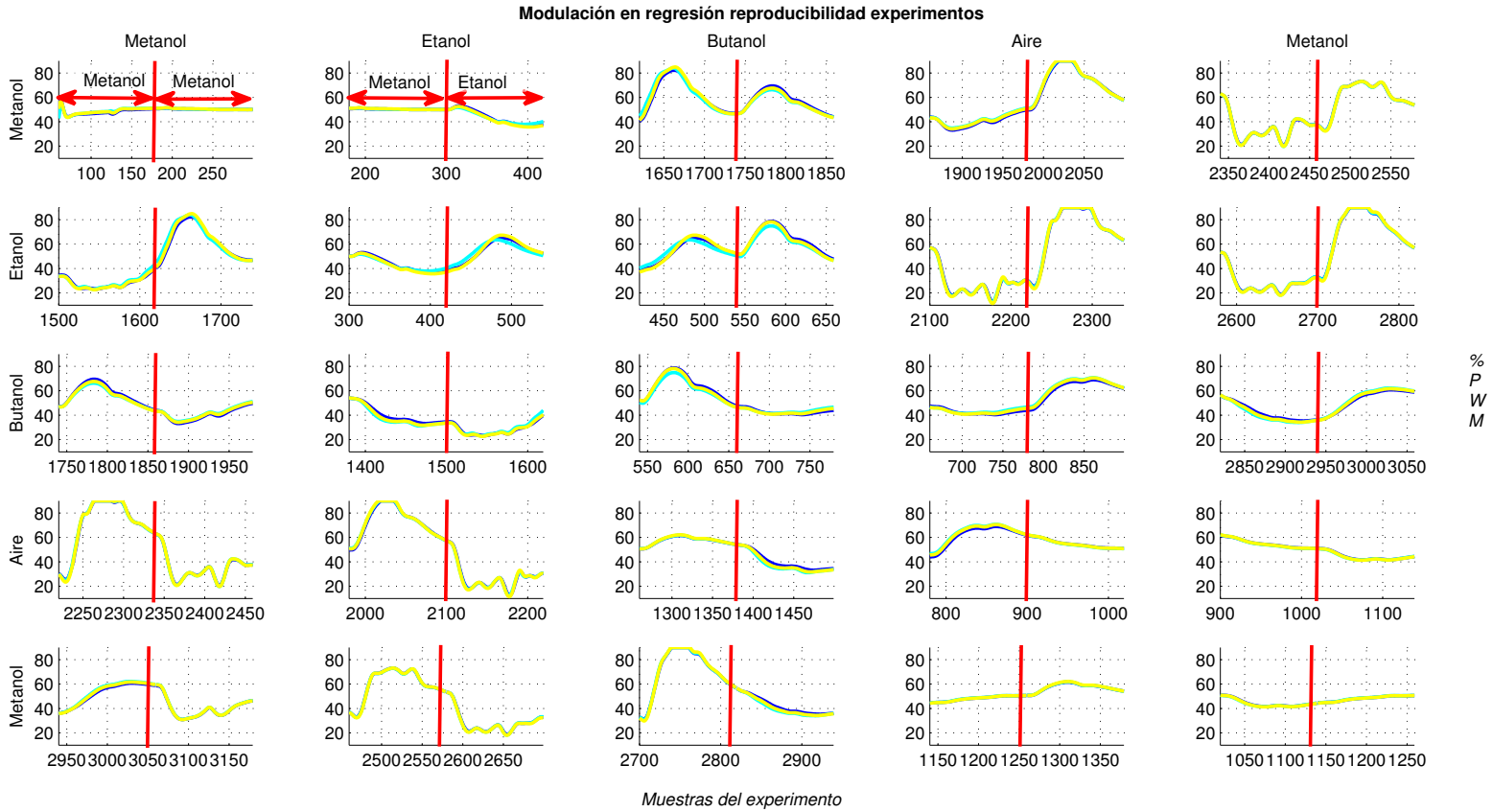



Figura 4.15: Experimento de reproducibilidad en la segunda placa de experimentación de la modulación en amplitud para 3 experimentos, en el que se representa temperatura aplicada al sensor. En la figura se representa en la fila el odorante en donde se está realizando actualmente la lectura y en la columna el odorante al que se va hacer la transición (cambio de odorante). Cada uno de los paneles se lee de la siguiente forma: por ejemplo, metanol(fila)-metanol(columna) quiere decir que hasta la línea roja es metanol y luego se hace la transición al propio metanol. Otro ejemplo es metanol-etanol quiere decir que hasta la línea roja es metanol y luego se hace la transición al etanol.

### 4.7.3. Modulación en frecuencia

La modulación en frecuencia es una de la modulaciones que fue implementada en la placa de experimentación construida en protoboard [5], y que ha sido integrada en la placa construida para este proyecto. En este tipo de modulación se varia la señal de temperatura basado en la frecuencia. Este método fue desarrollado por el profesor Martinelli [13], y se explica que el efecto que produce un odorantes en el sensor depende de la temperatura, y que cada odorante tiene una sensibilidad a la temperatura propia, todo esto basado en el concepto de "self-adapted temperature modulation".

Martinelli propone un circuito en lazo cerrado, en el que la salida del sistema es usada como la señal de entrada para el calentamiento del sensor. En la figura 4.16, se muestra el diagrama de bloques del sistema que plantea Martinelli.

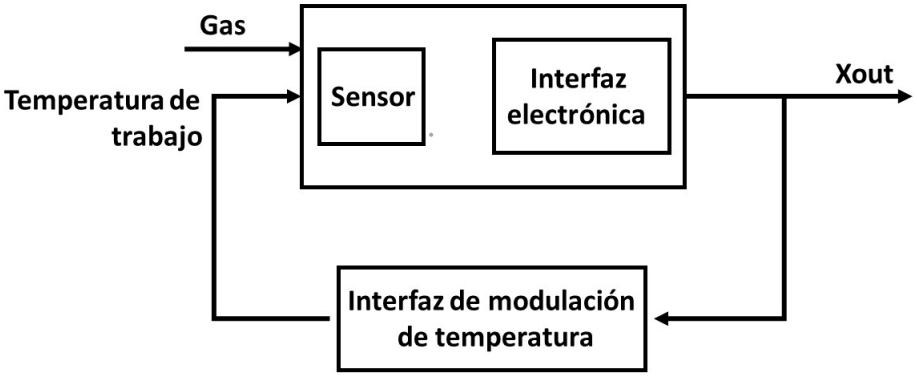



Figura 4.16: Diagrama de bloques del circuito en lazo cerrado que implementa "self-adapted temperature modulation". figura adaptada de [13].

El diagrama de bloques mostrado en la figura 4.16, esta basado en un circuito en lazo cerrado en el que un primer bloque esta formado por el sensor y interfaz electrónica, los cuales tiene como función el captar y procesar la señal respectivamente. El segundo bloque que es la interfaz de modulación de temperatura se encarga de procesar la señal de salida $X_{out}$ con el objetivo de obtener la señal de calentamiento adecuando esta señal a los rangos de tensión y frecuencia de trabajo.

El circuito usado para la interfaz electrónica es un multivibrador astable, el cual funciona como circuito oscilador que ayuda a la generación de una señal periódica que interactúa con la interfaz de control de temperatura. Este circuito oscilador se basa en el uso de un temporizador NE555 [26], este integrado se usa comúnmente como un generador de frecuencia de reloj. Por tanto, dentro del sistema este genera un señal periódica cuadrada en el que la duración del periodo dependerá de la resistencia del sensor. Esto quiere decir que cuando la resistencia del sensor cambia el periodo de los pulsos generados por el NE555 [26] también lo hace. En este caso cuanto mas grande es el valor de la resistencia, la oscilación del circuito es mas lenta y viceversa.

Por tanto, cuando se expone un odorante al sensor este hace que cambie la resistencia interna del sensor y en función del tamaño de esta resistencia se genera un frecuencia u otra. De esta forma lo que se pretende es que en presencia de un odorante se cuentan cuantos ciclos hay en frecuencia, en este caso se cuentan 8 ciclos que es un parámetro ya definido. Cada 8 ciclos que se cuenten se subirá la tensión de calentamiento y en los siguientes 8 ciclos se bajara esta tensión, de esta forma se dice que cada 16 ciclos se realiza una transición. Por ende, cada transición en función de la resistencia interna del sensor es diferente en tiempo lo que se traduce en frecuencias diferentes.

A continuación se va a comprobar que el circuito que se diseño para esta modulación funciona

correctamente. Para ello se usara el código desarrollado en [33] que permite que los experimentos sobre la placa para la modulación sean versátiles.

#### 4.7.3.1. Pasos para lanzar un experimento de modulación en frecuencia

A continuación se detallaran los pasos a seguir para poder lanzar un experimento usando el circuito de modulación en frecuencia. El procedimiento a seguir es el siguiente:

1. Comprobar que el sensor TGS2600 esta conectado en la placa de experimentación en el conector correspondiente para esta modulación, en este caso sobre el conector que tiene la serigrafía "TGS_FREQ" sobre la placa.

2. Revisar las conexiones que hay entre las placas del circuito de protección, placa de experimentación y la BBB. En la tabla 4.5 se encuentran los pines que se deben comprobar que esta conectados correctamente.

3. Revisar que el cable de Ethernet este conectado a la BBB.

4. Conectar la fuente de alimentación de 5 V a la BBB, y esperar hasta que esta este encendida y se puede acceder. Una vez dentro situarse sobre el directorio /home/debian/codigosNuevos que es donde se encuentra el código para lanzar el experimento.

5. A continuación conectar la fuente externa de 24 V a la placa de experimentación.

6. Una vez estén las dos placas encendidas, lo primero que se hará es apagar el pin de la BBB que controla la succión del motor ya que este pin esta encendido por defecto y por tanto el motor estará funcionado. Para apagar el pin correspondiente sera necesario ejecutar el siguiente comando:

   ```
   python3 inicio.py
   ```

7. Ahora se ejecutara el siguiente comando para llevar a cabo el experimento para la modulación en regresión de temperatura:

   ```
   python3 PyHuele.py script_martinelli.txt configuracion_plataforma.txt
   ```

   En donde "PyHuele.py" es el ejecutable para la realización del experimento y los archivos "script_martinelli.txt" y "configuracion_plataforma.txt", son ficheros de configuración de la modulación que se quiere realizar y que se detallan en la siguiente sección 4.7.3.2.

8. Una vez terminado el experimento se recomienda apagar la plataforma de experimentación si esta no va hacer usada. Para apagar la BBB se lanzara el siguiente comando:

   ```
   shutdown -h now
   ```

   Se debe esperar hasta que la placa se apague, esto se comprobara por que los LEDs de la BBB se apagarán.

9. Una vez paga la BBB es conveniente desconectar primero la fuente de alimentación de 24 V de la placa de experimentación y a continuación la fuente de alimentación de 5 V de la BBB.

#### 4.7.3.2. Ejemplo de experimento modulación en frecuencia

Para llevar a cabo un experimento con la modulación en frecuencia sera necesario llevar a cabo los pasos descritos en la sección 4.7.3.1 hasta el punto 7, ya que antes de lanzar el experimento sera necesario editar los ficheros de configuración que necesita el programa de modulación.

Por una parte se tiene el el fichero de configuración "script_martinelli.txt", el cual contiene la información del experimento que se va a lanzar. A continuación, se muestra el contenido de este fichero:

```
modulation: 3
number_experiments: 10
suction: 70*
duration_stimulation: 10*
initial_samples: 10*
number_samples: //
time_between_stimulus: 0*
name_file: martinelli_capturas*
name_folder: Martinelli*
experiment_version: 1*
martinelli_execution_mode: 2*
martinelli_temperature_mode: 1*
minimun_temperature_martinelli: 60*
maximun_temperature_martinelli: 100*
```

En este caso muchos de los parámetros de configuración son los mismos que en el fichero de configuración "script_regresion.txt", descrito en la sección 4.7.2.2. A continuación se explican los nuevos parámetros que encuentran en este nuevo fichero de configuración:

- martinelli_execution_mode, este parámetro se debe configurar cuando se quiere lanzar un experimento de modulación en frecuencia. Existen dos modos de ejecución:
  1. Con este valor se selecciona el modo de captura en el cual se usa una ventana de tiempo.
  2. Con este valor, se selecciona el modo de captura en el cual se usa como referencia las transiciones entre estado. Al usar este modo, el parámetro **duration_stimulation**, hace referencia a las transiciones entre los estados.
- martinell_temperature_mode, este parámetro se debe configurar cuando se quiere lanzar un experimento de modulación en frecuencia. El valor de este parámetro determina la forma en la que varía la temperatura del sensor, los posibles valores son:
  1. Con este valor el modo de cambio de la temperatura se hace de forma abrupta. Esto quiere decir, que la temperatura pasa del valor mínimo al máximo de golpe y viceversa.
  2. Con este valor el modo de cambio de la temperatura se hace de forma gradual. Por tanto, la temperatura ira cambiando de forma gradual entre el valor máximo y mínimo.
- minimun_temperature_martinelli, este parámetro representa la temperatura mínima que se usara en la modulación en frecuencia.
- maximun_temperature_martinelli, este parámetro representa la temperatura máxima que se usara en la modulación en frecuencia.

El siguiente fichero a configurar es "configuracion_plataforma.txt", el cual contiene la configuración de la placa y que en la sección 4.7.2.2 se han detallado los parámetros que se configuran en este fichero. A continuación, se muestra el contenido de este fichero para la primera como la segunda placa de experimentación respectivamente:

```
reading_port: P9_12
motor_pin: P9_21
resistance: 470
SD_folder: CAPTURAS/MARTINELLI
valve_position: R
heating_port: P9_14
sleep_time: 1
electrovalves_port: P8_10,P8_12,P8_14,P8_16
```

```
reading_port: P9_12
motor_pin: P9_21
resistance: 470
SD_folder: CAPTURAS/MARTINELLI
valve_position: R
heating_port: P9_14
sleep_time: 1
electrovalves_port: P8_10,P8_12,P8_14,P8_16,,P8_18
```

Como se puede observar, los parámetros que se han cambiado para el funcionamiento de la placa cuando se quiere lanzar un experimento de modulación en frecuencia son: reading_port, heating_port y SD_folder.

Una vez configurados los dos ficheros, se puede lanzar el experimento de la modulación en frecuencia. En este caso se van a llevar a cabo 10 experimentos en el cual la duración del estimulo en este caso es 10, que para esta modulación significa que habrá un cambio de odorante después de 10 transiciones. Cada transición se corresponde con 8 ciclos a la máxima temperatura y 8 ciclos a la mínima temperatura. Por otra parte, para este experimento se analizan todos los odorantes lo que conlleva a que se analicen todas las transiciones entre odorantes. El vector de electroválvulas de conmutación es el siguiente [[1], [1], [2], [2], [3], [3], [4], [4], [2], [1], [3], [1], [4], [3], [2], [4], [1]], siendo 1-Metanol, 2-Etanol, 3-Butanol y 4-Aire.

En la figura 4.17, se observa el resultado del experimento llevado a cabo para la modulación en frecuencia. En este caso se ha seleccionado uno de los 10 experimentos que se realizaron. En la figura 4.17, se representan todas las transiciones entre odorantes. La forma de interpretar esta figura es la siguiente, en cada fila se hace referencia al odorante en el que actualmente se esta realizando la lectura de sensor y en las columnas se encuentra el siguiente odorante sobre el que se realiza la transición. Por ejemplo, si se quiere ver la transición entre el butanol y el aire, se debe ir a la fila 3 y la columna 4.

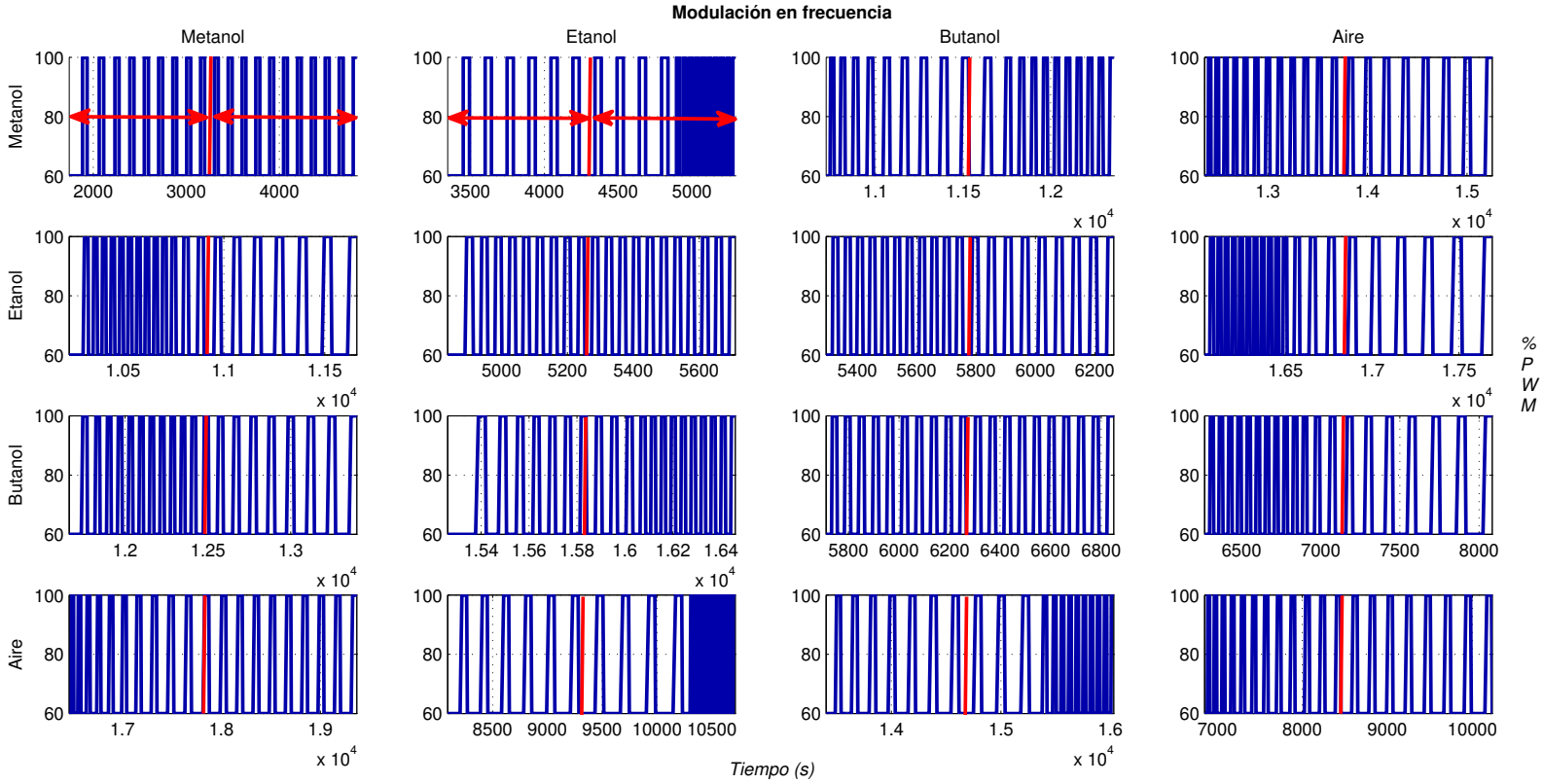



Figura 4.17: Prueba de la modulación en frecuencia sobre la primera placa de experimentación, en la que se representa los pulsos de oscilación de cada una de las transiciones entre odorantes de un experimento. La figura se lee de la siguiente forma: en la fila se encuentra el odorante en donde se está realizando actualmente la lectura y en la columna el odorante al que se va hacer la transición (cambio de odorante). Cada uno de los paneles se lee de la siguiente forma: por ejemplo, metanol(fila)-metanol(columna) quiere decir que hasta la línea roja es metanol y luego se hace la transición al propio metanol. Otro ejemplo es metanol-etanol quiere decir que hasta la línea roja es metanol y luego se hace la transición al etanol.

A simple vista resulta complicado ver lo que esta pasando durante el experimento. En este caso la modulación en frecuencia representa la variación en frecuencia de oscilación de cada uno de los odorantes al llevar a cabo el proceso de lectura. Cada odorante tiene una frecuencia de oscilación que lo caracteriza y distingue de los otros odorantes. Por ejemplo en la figura 4.17, si se observa la diagonal con los paneles vista de izquierda a derecha, cada panel representa las transiciones entre odorantes iguales. Esto quiere decir, Metanol-Metanol, Etanol-Etanol, Butanol-Butanol y Aire-Aire. Se puede observar como en estas transiciones la frecuencia de oscilación es constante. En otros paneles en las cuales hay cambio de odorante, se observa que la frecuencia de oscilación es distinta ya que cada odorante deja su propia huella.

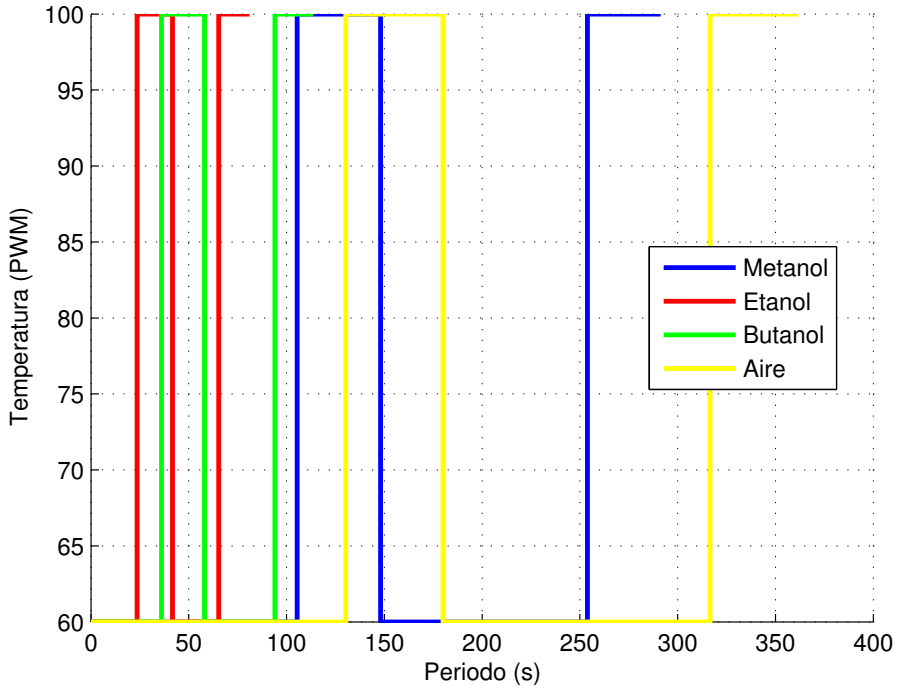



Figura 4.18: Frecuencia de oscilación de los odorantes bajo estudio para modulación en frecuencia.

Si se hace una ampliación de la figura 4.17 y se cogen muestras de cada uno de los odorantes bajo estudio (por ejemplo, tomar muestras de los paneles que forman la diagonal vista de izquierda a derecha), se puede observar la huella que deja cada odorante, esto quiere decir la frecuencia de oscilación que caracteriza al odorante. En la figura 4.18, se puede observar la frecuencia de oscilación de cada uno de los odorantes. En este caso se observa como para el etanol la frecuencia de oscilación es mayor con respecto a las concentraciones de metanol y butanol. Por otra parte, se observa como el aire tiene una menor frecuencia de oscilación.

Con este primer experimento, se comprueba como la modulación en frecuencia funciona en la placa desarrollada. Por tanto, el siguiente paso sera comprobar la reproducibilidad de los experimentos.

En el ejemplo del primer experimento para la modulación en frecuencia se configuro que el cambio de temperatura fuese de forma abrupta como se puede observar en la figura 4.18 en el que la temperatura solo cambia entre un 100 % y un 60 % del ciclo PWM. A continuación, se realiza un nuevo experimento cambiando el parámetro de configuración para que la temperatura cambie de forma gradual entre unos valores máximo y mínimo que también se configuran dentro del fichero de configuración, estos cambios se pueden apreciar a continuación:

```
modulation: 3
number_experiments: 10
suction: 70*
duration_stimulation: 5*
initial_samples: 5*
number_samples: //
time_between_stimulus: 0*
name_file: martinelli_capturas*
name_folder: Martinelli*
experiment_version: 1*
martinelli_execution_mode: 2*
martinelli_temperature_mode: 2*
minimun_temperature_martinelli: 60*
maximun_temperature_martinelli: 100*
```

En este caso, el fichero se configura para que el cambio de temperatura se haga de forma gradual entre una temperatura mínima de 60 % y una temperatura máxima de 100 % del ciclo PWM. Además se configura para que el numero de transiciones sean solo 5, lo cual hace que el tiempo total del experimento disminuya.

En este nuevo experimento al igual que el anterior la forma en la que se codifican los distintos odorantes, es midiendo el tiempo que se tarda en completar las transiciones, de tal forma que el odorante deja una huella representativa según el cambio de la temperatura en sus consecutivas transiciones. En este caso esta huella representativa es mas sencilla de ver gracias a que el cambio de temperatura es gradual. En la figura 4.19, se presentan las capturas de los odorantes para una transición entre estados. En cada panel se representa el tiempo medio del ancho de pulso en el que tarda en completar 5 transiciones en el paso de un odorante a otro.

En la figura 4.20, se puede observar el ciclo completo de cada uno de los odorantes completando las 5 transiciones, y en el que se observa como dependiendo del odorante desde el que se haga el cambio el tiempo en completar los ciclos es distinto para cada odorante y por tanto se puede distinguir cada odorante.

Tras probar la técnica de modulación sobre la primera placa de experimentación y comprobar que funciona correctamente se pasa a probar la modulación en frecuencia sobre la segunda placa. En este caso se decidió lanzar el ultimo experimento que se realizo con la primera placa en la que la temperatura cambia de forma gradual entre un valor máximo y mínimo. En este caso, la placa usa 5 odorantes y por tanto se realizara el experimento con todas las posibles transiciones usando el siguiente vector de conmutación: [[1], [1], [2], [2], [3], [3], [4], [4], [5], [5], [4], [3], [2], [1], [3], [1], [4], [2], [4], [1], [5], [2], [5], [3], [5], [1]], , siendo 1-Metanol, 2-Etanol, 3-Butanol, 4-Aire y 5-Metanol.

En la figura4.21, se observan los resultados obtenidos del experimento con la segunda placa en donde se muestra el tiempo que tarda en completar cada odorante 5 transiciones. Cada odorante tarda un tiempo distinto en completar las transiciones según desde el odorante anterior de donde se venga. En este caso cabe resaltar que hay dos odorantes iguales, metanol, pero que están ubicados en diferentes electroválvulas. El metanol se encuentra ubicado en la primera y ultima electroválvula, se esperaría que la respuesta en cada una de las transiciones fuese igual. Como se observa en este caso las dos muestras de metanol, llegan a presentar la misma huella para cada una de las transiciones, exceptuando el cambio de odorante entre el etanol y el metanol en que que se ve que el metanol1 y el metanol2 no presentan la misma huella, esto puede ser debido a que la frecuencia sea mas sensible en este cambio de odorante.

Tras realizar distintos experimentos con las dos placas se ha podido comprobar el funcionamiento de la técnica de modulación en frecuencia, en el que se ha observado como cada odorante

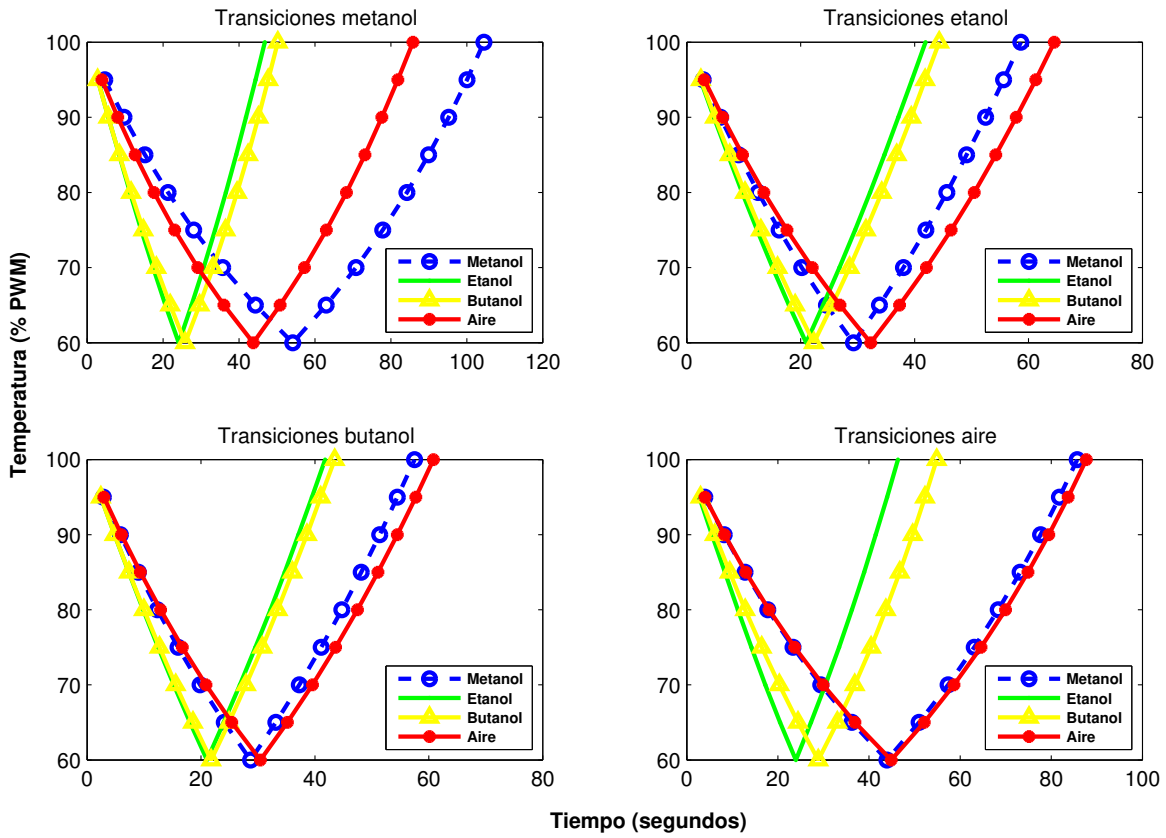



Figura 4.19: Representación del ancho de pulso medio de 10 experimentos para cada transición entre odorantes en la primera placa de experimentación. La figura se lee de la siguiente forma: en el titulo de cada panel se muestra el odorante desde el que se realiza el cambio de odorante. Por ejemplo, en el panel superior de la izquierda el titulo es "Transiciones metanol", esto quiere decir que estaba leyendo la respuesta del metanol y hago el cambio del metanol a los odorantes que se representan en cada panel.

tiene una huella característica representada por la frecuencia de oscilación con la que se consigue realizar un numero fijo de transiciones. Además es importante tener en cuenta que esta frecuencia de oscilación depende de factores como son la concentración del odorante y el odorante previo al odorante bajo estudio.

## 4.8. Estudio de variación de la temperatura en el sensor TGS2600

Como se menciono al inicio de este capitulo, se realizaría un estudio del comportamiento del sensor TGS2600 variando la temperatura de calentamiento, ya que el comportamiento de este sensor no siempre es el mismo y por tanto es importante ver cual es la respuesta del sensor ante cada una de las muestras con el cambio de temperatura, de esta forma se podrá tener un idea de como trabaja el sensor.

Este estudio se realiza llevando a cabo una lectura del sensor sin aplicar ningún tipo de modulación por tanto es una lectura pura del sensor. En este caso a diferencia de los experimentos que se realizaron en 4.7.1 lo que habrá es una variación en la temperatura aplicada al sensor. La idea ha sido crear una señal en forma de rampa en la cual habrá un cambio de temperatura gradual desde un valor máximo y un valor mínimo. Por tanto se tendrá una señal en la que por

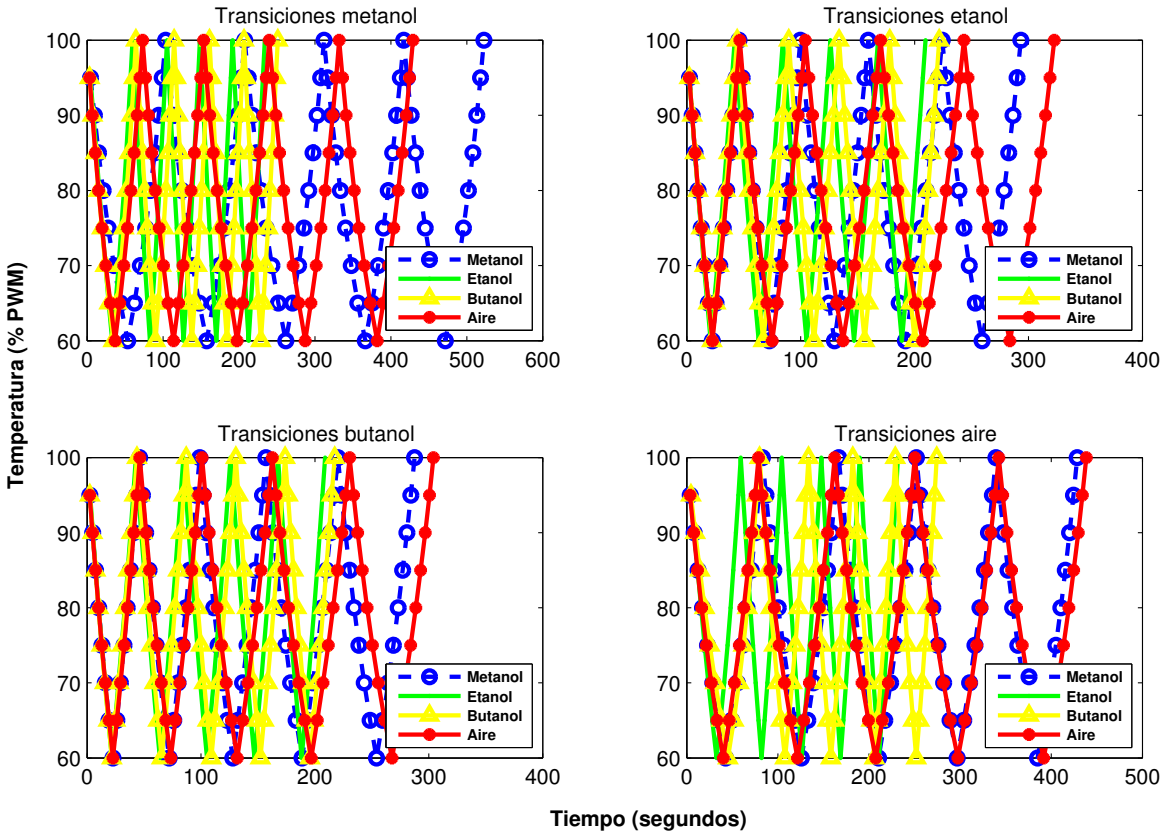



Figura 4.20: Representación media de la duración de las 5 transiciones de 10 experimentos en la primera placa de experimentación. La figura se lee de la siguiente forma: en el titulo de cada panel se muestra el odorante desde el que se realiza el cambio de odorante. Por ejemplo, en el panel superior de la izquierda el titulo es "Transiciones metanol", esto quiere decir que estaba leyendo la respuesta del metanol y hago el cambio del metanol a los odorantes que se representan en cada panel.

una parte habrá un ascenso y descenso de la temperatura y por otra parte en la temperatura máxima y mínima habrá un tiempo en el que esta temperatura se mantenga constante de esta forma se podra ver cual es el comportamiento del sensor con el cambio de temperatura.

Para llevar a cabo el estudio se seguirán los pasos descritos en 4.7.1.1 hasta el paso 7. Lo siguiente sera ejecutar el siguiente comando para llevar a cabo el experimento:

python3 rampa.py succión ciclos válvula

En donde succión, ciclos y válvula son los parámetros que el usuario debe introducir. El valor del parámetro succión debe ser un valor comprendido entre el 65 % y 100 %, este parámetro representa la potencia de succión del motor. Ciclos es el numero de veces que se repite el ciclo de la rampa de temperatura que se va aplicar al sensor TGS2600. Por el ultimo, el parámetro válvula representa la electroválvula del odorante que se quiere succionar.

Todos los experimentos que se han llevado acabo para ver el comportamiento del sensor ante cada odorante se han realizado con los mismos parámetros como son el la succión del motor al 70 % y el número de ciclos que ha sido 15. En el anexo I.3, se puede encontrar el código usado para este experimento.

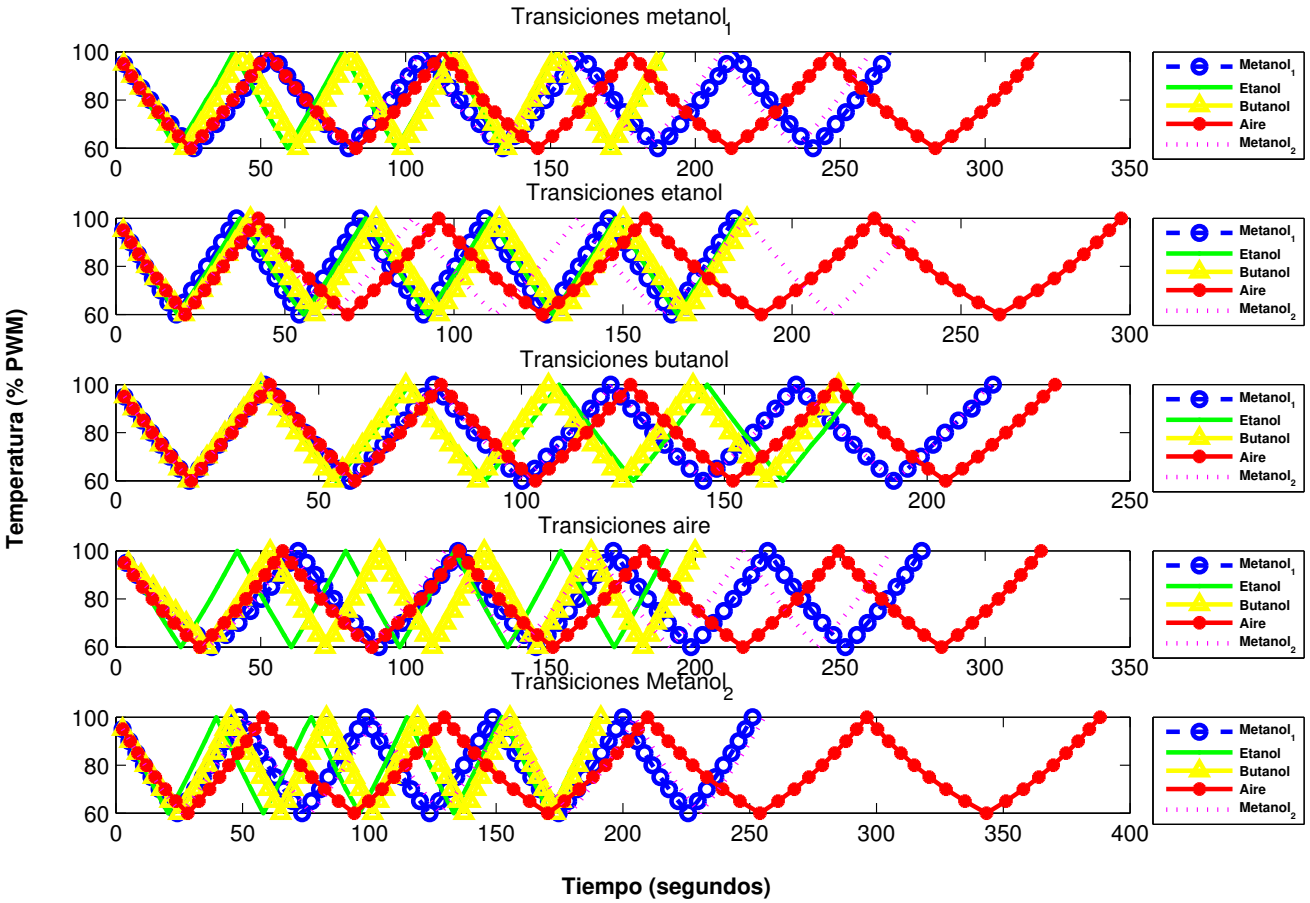



Figura 4.21: Representación media de la duración de las 5 transiciones de 6 experimentos en la segunda placa de experimentación. La figura se lee de la siguiente forma: en el titulo de cada panel se muestra el odorante desde el que se realiza el cambio de odorante. Por ejemplo, en el panel superior de la izquierda el titulo es "Transiciones metanol", esto quiere decir que estaba leyendo la respuesta del metanol y hago el cambio del metanol a los odorantes que se representan en cada panel.

En la figura 4.22, se representa la respuesta del sensor para cada uno de los odorantes en la primera placa de experimentación, tras aplicar la señal de temperatura de calentamiento en el proceso de lectura del sensor. Como se puede observar cada odorante tiene unos valores de respuesta diferentes ante el cambio de temperatura, que depende de la concentración del odorante. Cuando la señal de temperatura esta en proceso de ascenso se observa como a medida que la temperatura aumenta el valor de tensión de salida del sensor también lo hace, hasta cierto punto en el que se observa como aunque la temperatura siga aumentado el valor de salida disminuye abruptamente y se mantiene en algún caso estable como paso con el metanol y el etanol, mientras que para el butanol la lectura del sensor sigue cayendo aunque la temperatura se mantenga constante. Cuando la señal de temperatura esta en proceso de descenso se observa que la salida del sensor disminuye a medida que esta va descendiendo.

En la figura 4.23, se representa la respuesta del sensor para cada uno de los odorantes bajo estudio en la segunda placa de experimentación después de haber aplicado una señal de temperatura de calentamiento durante el proceso de lectura del sensor. Como se puede ver el sensor se comporta de forma similar a como sucedió en el experimento anterior sobre la primera placa de experimentación. Cuando la temperatura del sensor empieza a ascender el sensor lee

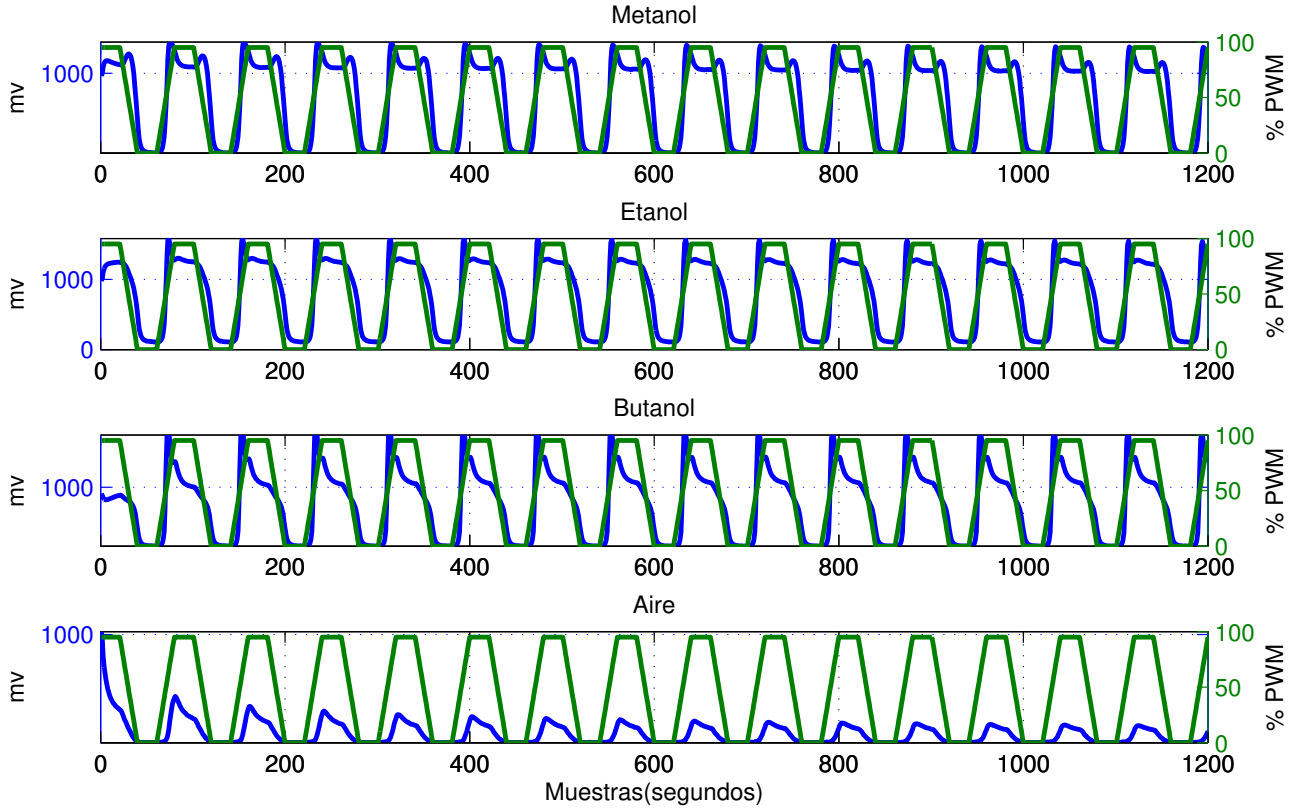



Figura 4.22: Respuesta del sensor TG2600 ante lo diferentes odorantes aplicando la señal de referencia de calentamiento en la primera placa de de experimentación.

valores de tensión que aumentan hasta cierto punto en el cual el sensor tiene una subida de tensión y aunque la temperatura aumente el valor de salida decrece, en este punto puede que el sensor se sature. cuando la temperatura comienza a descender la tensión de salida también lo hace. En este experimento al igual que el anterior se observa como cuando se mantiene constante la temperatura máxima durante varios segundos el valor del sensor no es constante si no que por el contrario este empieza a descender lentamente.

En el experimento realizado con la segunda placa de experimentación en el panel superior de la figura 4.23, se representa los valores leídos para el metanol el cual se encuentra ubicado en la primera y ultima electroválvula. Como se puede observar al superponer los valores leídos por el sensor se observa que la respuesta ante la rampa de temperatura aplicada es muy similar para cada odorante odorante.

El hecho de realizar estos experimentos es para ver el comportamiento del sensor TGS2600 con la variación de temperatura debido a que es importante conocer previamente como varia la lectura del sensor ante la temperatura que se aplica, ya que en el desarrollo de este trabajo el objetivo principal es aplicar una técnica de modulación mediante un control PID y por tanto esto hace que se tenga un conocimiento previo del funcionamiento del sensor. Es importante tener en cuenta que el sensor no tiene definido una relación entre la entrada y la salida, por tanto el poder observar previamente como se comporta el sensor ante el cambio de temperatura y ver la salida sirve como un primer acercamiento para poder predecir cual va a ser el posible comportamiento que tenga el sensor. En este caso se ha visto como los cambios de temperatura de forma instantánea hacen que la dinámica del sensor cambie de forma abrupta. Además se ha

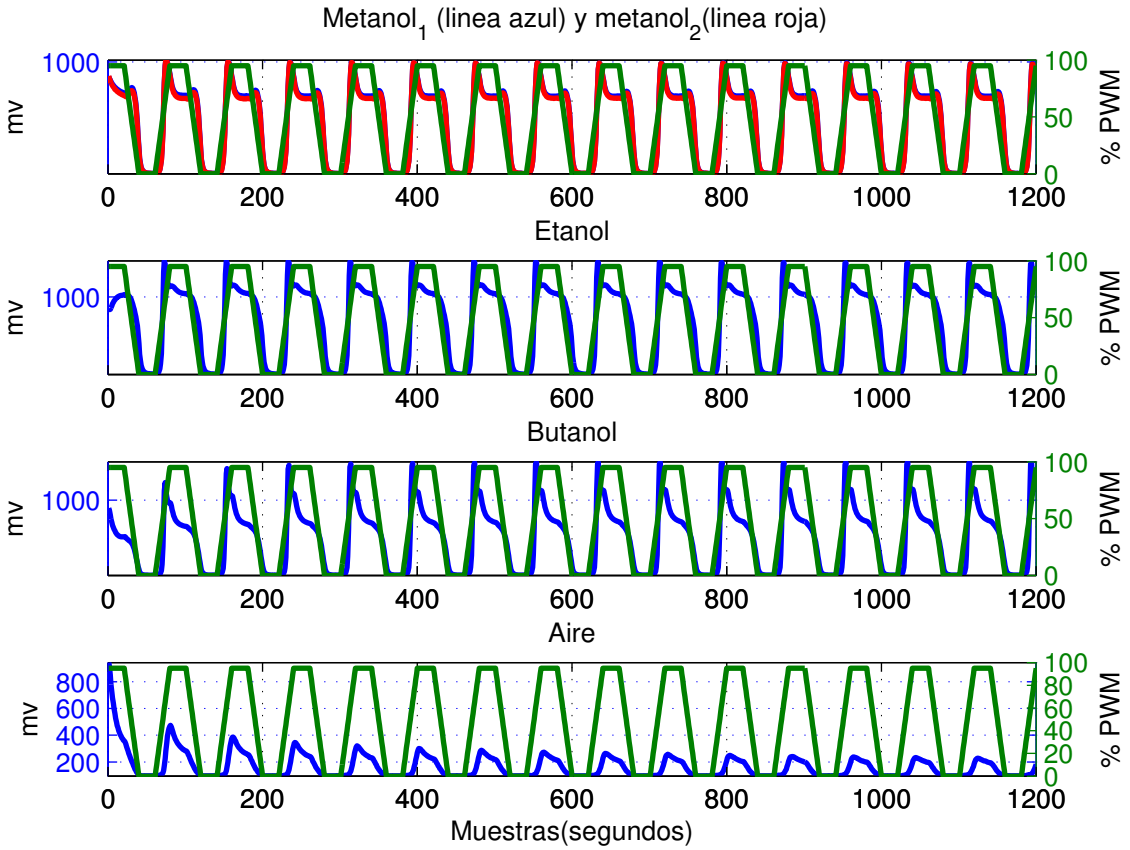



Figura 4.23: Respuesta del sensor TG2600 ante lo diferentes odorantes aplicando la señal de referencia de calentamiento en la segunda placa de de experimentación.

podido observar como con un temperatura baja el sensor en ningún momento llegar a tener lecturas iguales a 0mv, si no que por el contrario para todos los odorantes al aplicar la temperatura mínima el sensor lee valores de aproximadamente 100mV.

Por ultimo, aunque ya se ha mencionado que el sensor no tiene una relación entre la entrada y la salida, a la vista de los resultados en las figuras 4.22 y 4.23, se ha observado que hay una zona de trabajo en la que el sensor llega a trabajar de forma lineal y por tanto se podría encontrar la relación de la entrada y la salida linealizando en esta zona. Esto no quiere decir que el sensor trabaje de forma lineal en todo su rango y para cualquiera odorante de la misma forma pero si se podría llevar a cabo una aproximación.

# 5

# Implementación y pruebas de la modulación en lazo cerrado con un controlador PID

## 5.1. Introducción

El objetivo de este capítulo es poder describir el proceso que se ha llevado a cabo durante el desarrollo e implementación de la técnica de modulación en lazo cerrado con un controlador PID. La implementación de la técnica desarrollada tiene como punto de partida el trabajo realizado en [1], en el cual se desarrolla una estrategia de modulación en temperatura activa e inversa. La estrategia de modulación que se sigue es activa ya que depende de la respuesta que se obtiene del sensor en cada instante y es inversa porque primero se fija una señal de referencia y luego se calcula el valor de temperatura que se debe aplicar al sensor para poder obtener el valor deseado. Es importante tener en cuenta que la técnica que se ha desarrollado en este trabajo es distinta a la que se implementado en [1], aunque sigue la misma estrategia de modulación el control del sistema no es el mismo.

Al final del capitulo anterior en la sección 4.8 se realizo un estudio del comportamiento del sensor TGS2600 [6] ante la variación de la temperatura, lo cual permitió ver como un odorante no solo esta caracterizado por un único valor de conductividad si no que puede tener varios valores a distintas temperaturas de trabajo del sensor. Por tanto, partiendo de este estudio previo la técnica de modulación que se va a implementar explora los diferentes regímenes de comportamiento dinámico buscando ajustar la temperatura adecuada al odorante bajo estudio como la respuesta del sensor.

Una vez es implementada la técnica de modulación se pasará a realizar las pruebas sobre las plataformas de experimentación construidas para comprobar el funcionamiento de la técnica.

## 5.2. Desarrollo e implementación de la técnica de modulación

La técnica de modulación que se implementa en este trabajo se basa en una modulación activa e inversa de la temperatura usando un sistema de lazo cerrado con un controlador PID. La implementación de esta técnica sigue el esquema que se muestra en la figura 5.1.

En este esquema se lleva a cabo un proceso de sensado a través de un algoritmo de modulación de temperatura que tiene forma de una sinusoide. En este esquema el objetivo principal es buscar los valores mínimos y máximos del sensor con los que se pueda obtener datos característicos del odorante bajo estudio. En el esquema se tiene por una parte un lazo el cual controla al sensor para poder llevarlo a la señal de referencia usando un controlador PID, y por otra parte hay un lazo el cual actualiza la señal de referencia teniendo en cuanta la actividad del sensor.

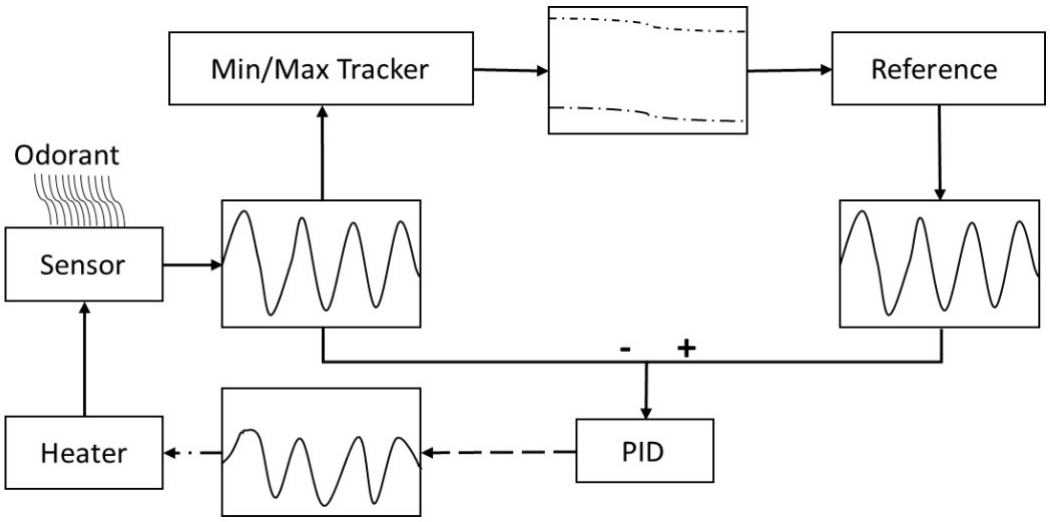



Figura 5.1: Esquema del proceso de la técnica de modulación. Figura adaptada de [1]

Por tanto el proceso de funcionamiento del esquema es el siguiente:

- El sensor responderá de forma diferente a cada uno de los odorantes bajo estudio, dependiendo de las condiciones de calentamiento de este mismo.
- El Tracker hace un seguimiento de los valores obtenidos por el sensor para obtener los valores máximos y mínimos.
- La señal de referencia es actualizada teniendo en cuenta los valores obtenido en el paso anterior, esta señal siempre conserva la misma frecuencia. Por tanto, en este paso puede que los valores del sensor sobrepasen o igualen los valores de de la referencia lo que hace que se module la referencia de forma especifica según el odorante bajo estudio.
- Un controlador PID, toma los valores de la señal referencia y los valores del sensor como entrada para calcular la temperatura que se debe aplicar al sensor para cada odorante bajo estudio.
- Los cambios producidos en la Heater hacen que cambien las propiedades fisicoquímicas del sensor bajo la presencia del odorante.

A partir del esquema explicado anteriormente se llevara acabo la implementación de la técnica de modulación. La implementación no tiene por que seguir el mismo proceso que el esquema de la figura 5.1, este simplemente es un esquema base que sirve de punto de partida para el desarrollo de la técnica.

La diferencia principal de la técnica que se ha desarrollado en este trabajo es principalmente la forma en la que se actualiza la señal de referencia, ya que en [1] la señal de referencia se deja evolucionar sin ejercer ningún tipo de control sobre la señal excepto el seguimiento que se hace

con el Tracker. En este trabajo se buscará explorar el rango dinámico del sensor ya que la idea es poder realizar un barrido en amplitud dependiendo de la temperatura máxima y mínima con la que se haya llegado a los valores máximo y mínimo de la señal de referencia, y no hasta donde indique el odorante bajo estudio.

Aunque la implementación de la técnica de modulación puede cambiar como se ha explicado previamente, hay dos parámetros que se seguirán usando al igual que se ha hecho en [1]. Estos parámetros son los siguientes:

1. La frecuencia de oscilación de la señal de referencia, este valor el cual no debe ser demasiado bajo ya que tomaría un tiempo mayor para obtener las medidas ni tampoco puede ser alto puesto que el sensor podría no ser capaz de responder de forma inmediata lo que causaría perdida de información. Por tanto, la frecuencia de oscilación de la señal de referencia es de 50 segundos.

2. Constantes del controlador PID, el sistema implementa un lazo cerrado usando un controlador PID el cual se encarga de calcular la temperatura que se debe aplicar al sensor para conseguir el valor deseado. En el caso de este sistema se debe recurrir al uso de las reglas de Ziegle-Nichols para seleccionar los parámetros del controlador PID, ya que el sensor no tiene un modelo matemático el cual defina la planta. En el anexo G, se puede encontrar la aplicación de uno de los métodos que proponen Ziegle-Nichols para hallar los parámetros del controlador. En la tabla 5.1 se muestran los valores de las constantes del controlador PID que se usarán en el desarrollo del proyecto.

| **Constante** | **Valor** |
|---|---|
| $K_p$ | $0{,}6 \cdot 0{,}3 = 0{,}18$ |
| $K_d$ | $50/8 = 6{,}25$ |
| $K_i$ | $50/2 = 25$ |

Tabla 5.1: Valores de las constantes del controlador PID usados en [1].

Partiendo de lo descrito anteriormente en la figura 5.2, se muestra el proceso que se ha seguido para llevar a cabo el desarrollo de la técnica de modulación. De izquierda a derecha se muestra cada una de las partes de proceso durante lo que sería la ejecución de un experimento. Este proceso se ha divido en 3 etapas que se explicaran detalladamente. Durante el desarrollo de cada una de las etapas se mostrara parte del código de la implementación de algunos de los procesos. El código completo de la técnica de modulación se podrá encontrar en los anexos I.5.2 y I.5.3, para la primera y la segunda placa de experimentación respectivamente.

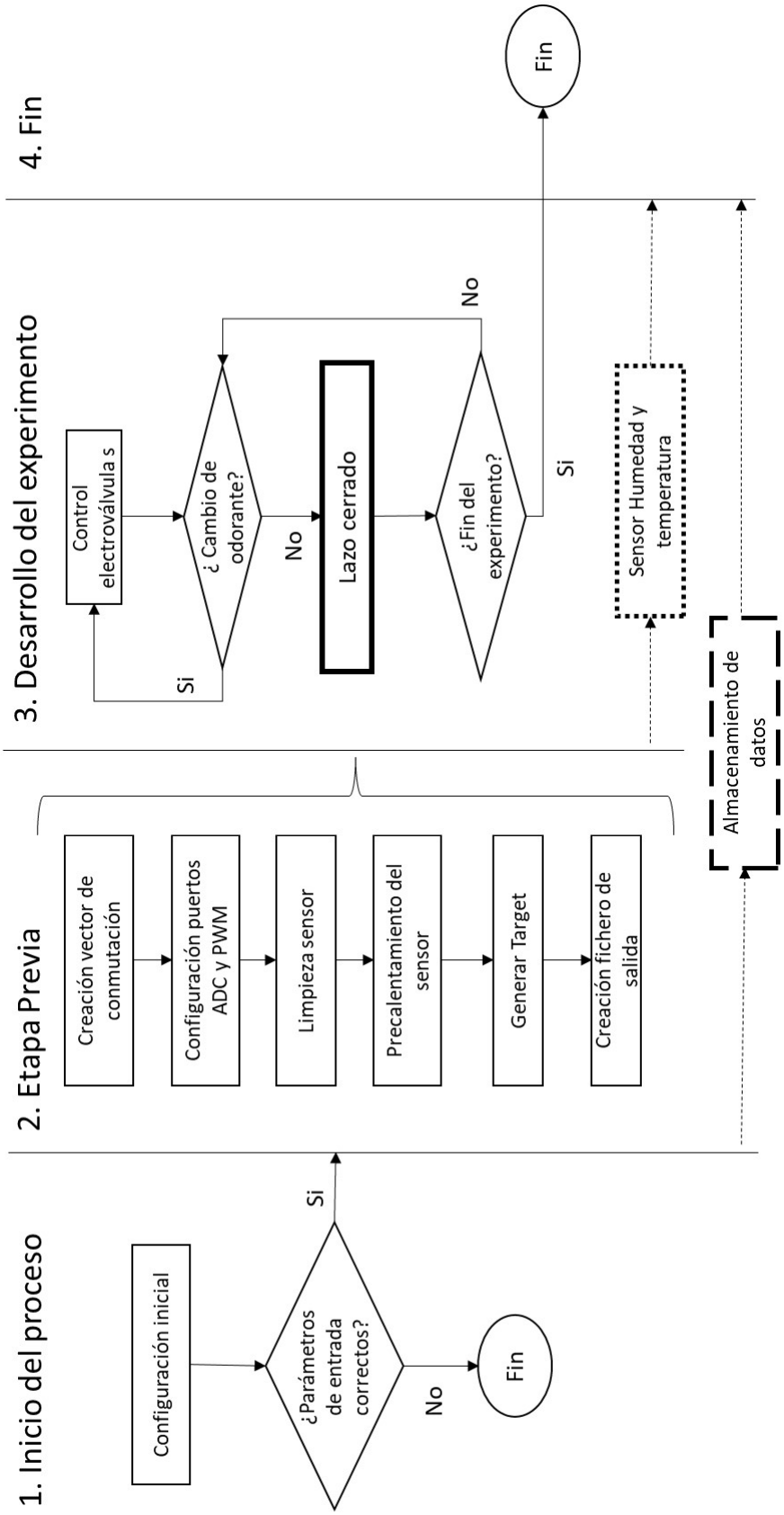



Figura 5.2: Esquema de la técnica de modulación en lazo cerrado

### 5.2.1. Inicio del proceso

Esta es la etapa inicial del proceso de ejecución de un experimento, ver figura 5.2. En esta etapa el objetivo principal es importar los paquetes necesarios para la realización del experimento, declaración de las variables del sistema y parte de la configuración de los puertos de la placa BBB.

#### 5.2.1.1. Importación de paquetes para el sistema

A continuación se muestra los paquetes que se deben importar y que son prioritarios para el sistema. En este caso como se observa es necesario importar los paquetes para llevar a cabo el control de los puertos GPIO, puertos PWM y ADC, así como un paquete especial para el uso del sensor de temperatura y humedad DTH22. El proceso de instalación de estos paquetes se puede encontrar en el anexo B.

```
import Adafruit_BBIO.ADC as ADC
import Adafruit_BBIO.GPIO as GPIO
import Adafruit_BBIO.PWM as PWM
import Adafruit_DHT
import sys
import os
import time
import math
import random
import numpy
from datetime import datetime, date
import _thread
from math import ceil
from  math import sin
from  math import pi
```

#### 5.2.1.2. Definición de variables global del sistema

Por una parte, se tiene las variables del sistema que no cambiaran durante toda el proceso de ejecución del experimento, y variables que que si que pueden que tomen distintos valores durante el experimento. En la tabla 5.2 se definen aquellas variables del sistema que van a cambiar durante la ejecución del experimento y en la tabla 5.3 se describen las constantes del sistema.

| Variable | Descripción |
|---|---|
| ***target*** | Almacena los valore de la señal de referencia, esta variable se actualiza durante todo el experimento. |
| ***muestrasT GS*** | Almacena los valores leídos por el sensor |
| ***temperaturaTGS*** | Almacena los valores de ciclo PWM que se aplica para calentar el sensor en cada instante. |
| ***rS_ TGS*** | Almacena el valor de la resistencia del sensor en cada instante |

Tabla 5.2: Definición de variables del sistema que se actualizan durante la ejecución del experimento.

$$R_S = \frac{V_C \cdot R_L}{V_{OUT}} - R_L. \tag{5.1}$$

| Constante | Descripción | Valor |
|---|---|---|
| ***SLEEP*** | Determina el tiempo de muestreo en que se llevara a cabo la lectura del sensor. | 1 s |
| ***periodo*** | Periodo de la señal sinusoidal, con el cual se genera la señal de referencia. | 50 s |
| ***errorControl*** | Constante de control usada para controlar el proceso de Tracker y determinar si se debe actualizar la señal de referencia. | 5 % |
| ***temperaturaMin y temperaturaMax*** | Constantes que determina el valor mínimo y máximo del ciclo PWM ( tensión de calentamiento) que se aplica al sensor. | 5 % y 95 % |
| ***tempMinMinControl y tempMinMaxControl*** | Constantes usadas para el proceso de actualización de la señal de referencia respectoal valor mínimo, teniendo en cuenta el ciclo PWM ( tensión de calentamiento). | 5 % y 15 % |
| ***tempMaxMinControl y tempMaxMaxControl*** | Constantes usadas para el proceso de actualización de la señal de referencia respecto al valor máximo, teniendo en cuenta el ciclo PWM ( tensión de calentamiento). | 95 % y 85 % |
| ***numCiclos*** | Constante usada para determinar el numero de ciclos de precalentamiento del sensor antes de la realización del experimento. | 1 |
| ***Rl, Vc*** | Resistencia de carga y tensión de alimentación del sensor, respectivamente. Estas dos constantes son usadas para el calculo de la resistencia interna del sensor ecuación 5.1. El valor de la resistencia Rl es un parámetro que se puede cambiar a nivel de HW | 470Ω y $5V$ |
| ***Kp, Kd y Ki*** | Constantes del controlador PID | Ver tabla 5.1 |

Tabla 5.3: Definición de las constantes del sistema.

Por ultimo, queda por definir aquellas variables a las cuales se les asignan los valores de los puertosde la BBB. En la tabla 5.4 se muestra las definición de la asignación de los puertos que se usaran durante la ejecución del experimento. Como se puede observar se asignan los puertos usados para el control de las electroválvulas, el calentamiento y lectura del sensor TGS2600, la lectura del sensor de temperatura y humedad y el control del motor.

#### 5.2.1.3. Parámetros de entrada del sistema

Una vez se han declarado las variables globales que se usaran en la ejecución del experimento, el siguiente paso durante el proceso es la comprobación de los parámetros de entrada para la ejecución del experimento. En este punto, se definirán los parámetros que el usuario debe introducir en el momento de realizar un experimento. Estos parámetros son los siguientes:

- *succion*, este parámetro se corresponde con la potencia de succión (ciclo PWM) que se aplicara al motor. El valor de este parámetro debe ser un valor entre 65 y 100, ya que por debajo de un 65 % del ciclo PWM no se consigue llegar a la potencia mínima que requiere el motor para trabajar y el valor máximo que se puede aplicar es del 100 %.

| Puerto | Variable | Código | Observación |
|---|---|---|---|
| P8_10 | electrovalve1 | electrovalve1 = 'P8_10' | Electroválvula 1 (Metanol) |
| P8_12 | electrovalve2 | electrovalve2 = 'P8_12' | Electroválvula 2 (Etanol) |
| P8_14 | electrovalve3 | electrovalve3 = 'P8_14' | Electroválvula 3 (Butanol) |
| P8_16 | electrovalve4 | electrovalve4 = 'P8_16' | Electroválvula 4 (Aire) |
| P8_18 | electrovalve5 | electrovalve5 = 'P8_18' | Electroválvula 5 (Metanol)* |
| P8_11 | Temp22 | Temp22 = 'P8_11' | Sensor de temperatura y humedad |
| P9_21 | motorPin | motorPin = 'P9_21' | Motor de succión |
| P9_22 | heatPin2600 | heatPin2600 = 'P9_22' | Heater TGS2600 |
| P9_40 | sensorPin2600 | sensorPin2600 = 'P9_40' | Lectura analógica del sensor |

Tabla 5.4: Definición de variables asignadoles los puertos usados en el experimento. * Electroválvula disponible solo para la segunda versión de la placa de experimentación.

- *heat*2600, este parámetro se corresponde a la temperatura inicial que se aplicará al sensor cuando se inicie la ejecución del experimento. Este parámetro puede tomar un valor entre el 0 % y el 100 % que se corresponderá con la tensión de calentamiento que se aplique al sensor.
- *tiempo*, este parámetro se corresponde con el tiempo que dura el experimento en minutos y el valor introducido debe ser mayor o igual que 1.
- *cambio*, este parámetro se corresponde con el tiempo de conmutación en segundos entre las electroválvulas.
- *odorante*, este parámetro se corresponde con el valor de la electroválvula que se desea abrir para realizar el proceso de calentamiento previo del sensor, antes de que se empiece el experimento. En este caso los valores que se pueden introducir son 1,2,3,4 y 5 (valor válido solo para cuando se use la segunda placa de experimentación).
- $maximoReferencia$, este parámetro se corresponde con el valor máximo de la señal de referencia que se va a crear y que el usuario puede introducir por parámetro.
- $minimoReferencia$ este parámetro se corresponde con el valor mínimo de la señal de referencia que se va a crear y que el usuario puede introducir por parámetro.

Una vez que se han introducido los parámetros de entrada que previamente se han descrito si estos son correctos se podrá pasar a la siguiente etapa para continuar con la ejecución del experimento, si no son correctos el experimento no podrá ser llevado a cabo.

### 5.2.2. Etapa previa

Esta es la etapa previa a la ejecución del experimento, ver figura 5.2. En esta etapa se llevan a cabo varios procesos secuenciales en los cuales se ponen en marcha algunos circuitos que componen la placa de experimentación.

En primer lugar en esta etapa se define el vector de conmutación, en este caso este contendrá todas las posibles transiciones entre los odorantes teniendo en cuenta que hay dos posibles vectores dependiendo de la placa de experimentación que se use:

- Primera placa de experimentación, el vector de conmutación es el siguiente: [[1], [1], [2], [2], [3], [3], [4], [4], [2], [1], [3], [1], [4], [3], [2], [4], [1]].

- Segunda placa de experimentación, el vector de conmutación es el siguiente: [[1], [1], [2], [2], [3], [3], [4], [4], [5], [5], [4], [3], [2], [1],[3], [1], [4], [2], [4], [1], [5], [2], [5], [3] ,[5], [1]].

Siendo 1-Metanol, 2-Etanol, 3-Butanol, 4-Aire y 5-Metanol, que serán las muestras de odorantes que se van a usar con las mismas concentraciones descritas en la tabla 4.3.

Una ves definido el vector de conmutación, el siguiente paso que se realiza el la configuración de los puerto PWM y ADC para su utilización. El código de configuración de estos puertos es el siguiente:

```
## Se configura los puertos ADC, PWM
motorStart(succion)
ADC.setup()
PWM.start(heatPin2600,heat2600)
```

A partir de este punto, el motor ya estará configurado para succionar con un ciclo de potencia que sera el que se haya pasado a la entrada del programa. Los puertos ADC estarán listos para ser leídos y al sensor TGS2600 ya se le puede aplicar el porcentaje de temperatura que se desee que inicialmente sera el que se haya pasado como parámetro de entrada.

#### 5.2.2.1. Limpieza y precalentamiento del sensor

Una vez realizada la configuración de los puertos PWM y ADC de la BBB, la primera acción que realizara el sistema sera un proceso de limpieza del sensor el cual consiste en capturar muestras del sensor con el objetivo de limpiar la posibles impurezas que hayan dentro del habitáculo del sensor, en este caso este proceso de limpieza tiene una duración de 30 segundo. a continuación se muestra el código encargado de realizar el proceso de succión.

```
samples = 0
while samples < 30:
  tick = time.time()
  print ('Medidas de estabilizacion:\t'+str(readADC()))
  samples+=1
  tack = time.time();
  time.sleep(SLEEP-(tack-tick))
```

Tras realizar el proceso de limpieza del sensor se pasa a llevar a cabo el precalentamiento del sensor. Este precalentamiento consiste en generar una señal con distintos valores de temperatura para aplicar al sensor mientras se esta succionando un odorante en concreto. La señal que se aplica a sensor es la misma señal que se ha usado en 4.8, en este caso el precalentamiento del sensor se realiza durante un ciclo que se corresponde a 80 segundos. Para realizar el precalentamiento se debe especificar con que odorante se va a realizar el precalentamiento en este caso el odorante se pasa como parámetro de entrada del sistema en la variable *odorante*.

#### 5.2.2.2. Generación señal de referencia

Antes de empezar con el experimento es necesario tener la señal de referencia de partida y que el sensor en el primer periodo va a ser capaz de seguir, después de este primer ciclo esta señal de referencia se actualizara de tal forma que la señal de referencia es la que se ira modulando para así poder explorar los regímenes del comportamiento dinámico al ajustar la temperatura que se aplicará al sensor.

La señal de referencia es una señal sinusoidal que tiene un periodo que se mantendrá estable a lo largo del experimento. En este caso, el periodo de la señal sera de 50 segundos, como ya se había definido previamente y que es el periodo usado en [1]. Por otra parte, para la generación de esta señal sera necesario pasar como parámetros los valores máximo y mínimo que tendrá la señal, los cuales son parámetros de entrada del programa principal. La función *crearTarget* es la encargada de generar los valores de la señal de referencia y se muestra a continuación:

```
def crearTarget(Vmax, Vmin, periodo):
    global target
    print('maximo:\t'+str(Vmax)+'\t minimo:\t'+str(Vmin))
    """ Asen(BX+C)"""
    A = (Vmax - Vmin)/2.0
    B = (2*pi)/(periodo) ## periodo

    senal =[]

    for x in range(periodo+1): ## periodo+1
      senal.append(round(((A*sin(B*x))+(A+Vmin)),3))

    target = senal[0:periodo] ## senal[1:periodo]
    print (target)
```

A la salida la función señal devolverá una array con los valores de la señal de referencia. Esta función es usada para la actualización de la señal de referencia cada vez que haya cambios en los valores del maximo y minimo de la señal y que se explicara mas adelante.

### 5.2.3. Desarrollo Experimento

En esta etapa del proceso es donde se lleva a cabo la realización del experimento, ver figura 5.2. Aquí se explicara la implementación y desarrollo de la técnica de modulación.

#### 5.2.3.1. Control electroválvulas

El control de electroválvulas es el proceso en el cual se controla que electroválvula debe estar abierta en cada momento del experimento, para llevar a cabo este control es necesario tener en cuenta el parámetro de entrada *cambio* el cual representa el tiempo de conmutación de cada electroválvula.

Dentro del sistema se hará la llamada a la función $abrirElectrovalvula(electrovalvula)$, ver sección 4.5. Esta función tiene como parámetro de entrada la electroválvula que se debe abrir para llevar a cabo el proceso de succión. En este caso la electroválvula que se deberá abrir vendrá determinado por la variable $vectorValvulas$ la cual contiene el vector completo de conmutación de electroválvulas del experimento el cual se ha definido previamente.

#### 5.2.3.2. Sensor humedad y temperatura

Además de tomarse medidas del sensor TGS2600, las placas de experimentación están preparadas para también ser capaces de recoger datos de temperatura y humedad del ambiente en el que se realiza el experimento. Esta lectura se llevara a cabo en un segunda plano durante toda la ejecución del experimento. La función usada para este proceso se llama $sensorTyH(tiempo)$, y tiene como parámetro de entrada el tiempo que durara la lectura, en este caso el parámetro de entrada que se le pasa es el parámetro de entrada del sistema *tiempo*. El código de la función

se puede encontrar en el anexo I.1. A continuación, se muestra el código usado para hacer la llamada a la función de lectura de el sensor DTH22 que hace que el proceso se ejecute en un segundo plano.

```
    _thread.start_new_thread(sensorTyH,(tiempo,))
```

#### 5.2.3.3. Control del sistema

En esta etapa es donde se lleva a cabo la implementación de la técnica de modulación, y en la que se detallara todo el proceso desde la captura de la muestra y como se determina cual debe ser la tensión de calentamiento que se debería aplicar al sensor para conseguir llegar a la señal de referencia deseada. En la figura 5.3, se muestra tres etapas del proceso del sistema de control por una parte se tiene la captura de la muestra y la exploración del máximo y el mínimo de la señal, a continuación se realiza el proceso del controlador PID y la actualización de la tensión de calentamiento del sensor y por ultimo se lleva a cabo el almacenamiento de datos y la actualización de la señal de referencia.

La llamada al sistema de control desde el programa principal es la siguiente:

```
1 datos = controlSistema(contador, subtarget,lastError,lastTime,addError,temp,
    flagTarget)
```

Los parámetros de entrada que se la pasan son los siguientes:

- *contador*, que es el contador del programa principal e indica el numero de muestra del experimento. Este parámetro se usa en el momento de almacenar los datos.
- *subtarget*, este parámetro contiene el valor de la muestra de la señal de referencia en cada instantes de tiempo.
- *lastError*, este parámetro contiene el error del ciclo anterior después de aplicar el control PID. El valor por defecto de esta variable es 0.
- *lastTime*, este parámetro contiene la ultima vez que se realiza la captura de un dato del sensor.
- *addError*, este parámetro contiene el error acumulado durante el experimento. El valor por defecto de esta variable es 0.
- *temp*, contiene el ultimo valor de temperatura que se ha aplicado al sensor TGS2600. Por defecto al inicio del proceso esta variable tomar el valor del parámetro de entrada que se recoge en la variable *heat*2600.
- *flagTarget*, este parámetro sirve para saber en que momento del experimento se debe realizar la actualización de la señal de referencia. Esta actualización de la señal se debe realizar cada vez que termine el periodo de la señal referencia, esto quiere decir cada ciclo de 50 segundos.

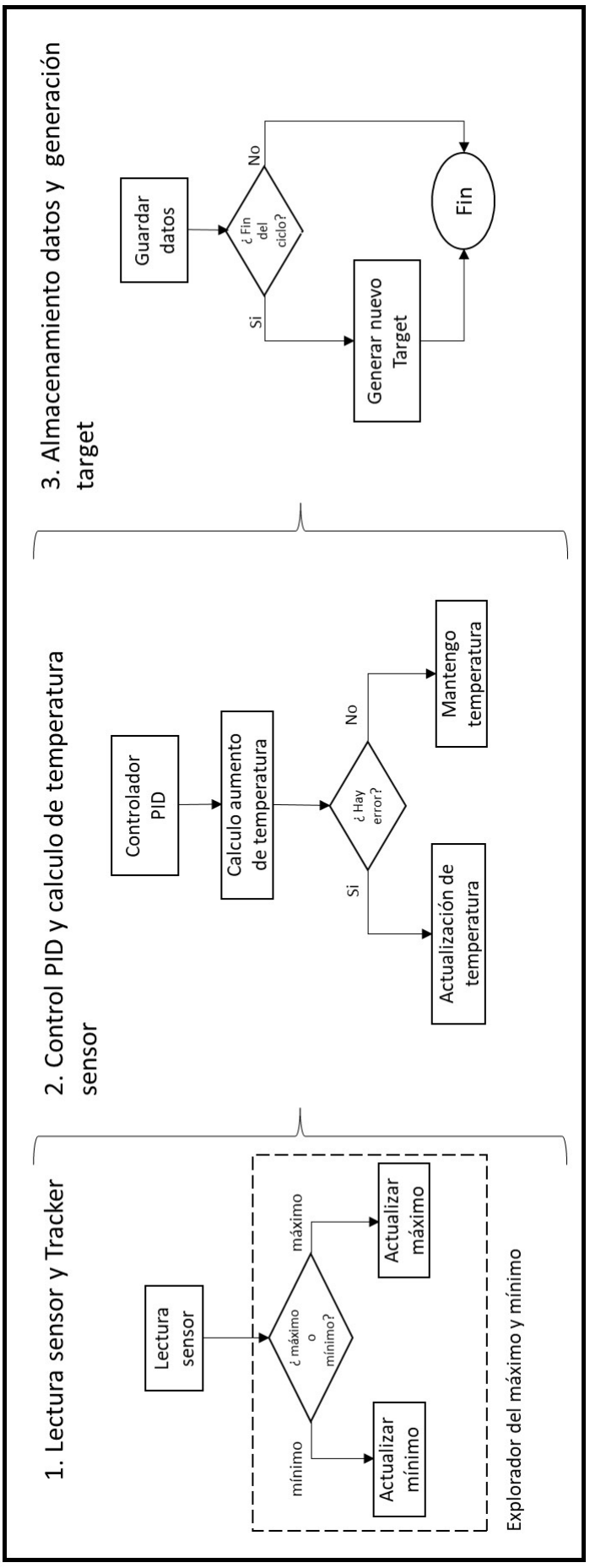



Figura 5.3: Esquema de la técnica de modulación en lazo cerrado

Por ultimo, la llamada a esta función devuelve un vector de salida *datos*, el cual contiene los valores de actualización de algunas de las variables de entrada de la función *controlSistema* como son *lastError*, *lastTime*, *addError* y *temp*.

**Lectura sensor y tracker** En esta primera etapa del sistema de control lo que se hace es realizar la lectura del sensor, esta lectura se hará una vez por segundo. Para realizar la lectura del sensor se hace la llamada a la función readADC, su implementación se muestra a continuación:

```
def readADC():
    valor = (1800*(ADC.read(sensorPin2600)))
    valor = round(valor,3)
    return valor
```

Para obtener el valor de lectura del sensor, se hace uso de la función read del paquete importado ADC, al cual se le pasa como parámetro el puerto analógico y este devuelve un valor normalizado entre 0-1,0. El valor devuelto es multiplicado por 1800 para así obtener la lectura real del sensor.

Una vez realizada la lectura del sensor, lo siguiente que se hace es comprobar si el parámetro de entrada *subtarget* de la función controlSistema, se corresponde con el valor máximo o mínimo de la señal de referencia para así poder comprobar si el máximo o el mínimo han cambiado de tal forma que hayan sobrepasado o estén por debajo de los valores previos de máximo y mínimo de la señal de referencia del ciclo actual.

Una de las constantes que se definió en 5.2.1.2 fue *errorControl*. Esta constante es usada para encontrar un rango entre el cual se puede aceptar el error del proceso del controlador PID, ya que no se tiene como objetivo que el sensor sea capaz de seguir la señal de referencia de forma perfecta si no que la señal de referencia pueda servir para encontrar el rango dinámico del sensor bajo la presencia del odorante.

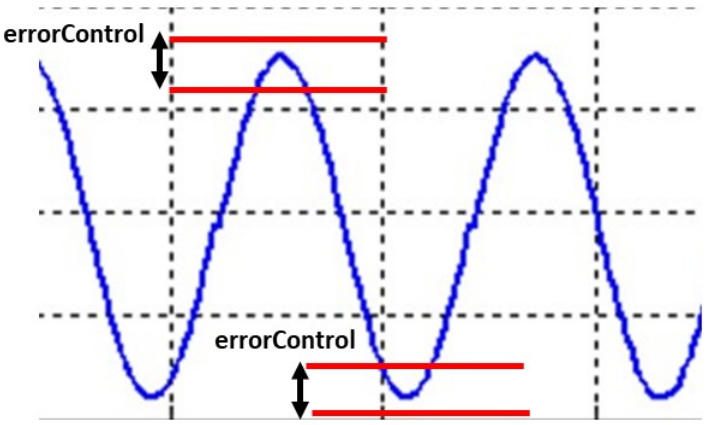



Figura 5.4: Control del error de la señal de referencia para la actualización de la señal de referencia

Por tanto, la actualización del valor máximo o mínimo se hará cuando el valor leído por el sensor sobrepase o este por debajo del rango del valor máximo o mínimo de la señal de referencia como se puede observar en la figura 5.4, esto quiere decir que pueden haber ciclos en los que la señal de referencia puede que no cambie su amplitud si el valor leído por el sensor en ese instante se encuentra dentro de estos rangos. El código con el cual se implementa el seguimiento de la señal es el siguiente:

```
## actualizo los valores maximos y minimos de la senal
```

```
  if subtarget == max(target):
    flagMaximo  = True;
    print ('Entro al maximo '+ str(flagMaximo))
    e = subtarget*errorControl;
    print ('error:\t'+str(e))
    if not(value>=(subtarget-e) and value<=(subtarget+e)):
      print ('El maximo cambia')
      maximo = value
    else:
      maximo = max(target)
  if subtarget == min(target):
    flagMinimo= True
    print ('Entro al minimo ' +str(flagMinimo))
    e = subtarget*errorControl;
    print('error:\t'+str(e))
    if not(value>=(subtarget-e) and value<=(subtarget+e)):
      print ('El minimo cambia')
      minimo = value
    else:
      minimo = min(target)
```

**Controlador PID** Una vez realizada la lectura del sensor y la exploración del máximo y el mínimo de la señal, el siguiente paso es usar un controlador PID para poder calcular la temperatura necesaria para que el sensor sea capaz de seguir la señal de referencia.

Lo primero que se va a hacer es recordar el funcionamiento del controlador PID partiendo de las ecuaciones 5.2 - 5.5

$$e_p = referencia - lecturaSensor \tag{5.2}$$

$$e_d = \triangle e_p \tag{5.3}$$

$$\triangle e_i = e_p \tag{5.4}$$

$$\triangle u = K_p(e_p + K_d \cdot e_d + \frac{1}{K_i} \cdot e_i) \tag{5.5}$$

En donde $e_p$ es el error proporcional, $e_d$ es el error derivativo, $e_i$ es la integral del error,$K_p$,$K_d$ y $K_i$ son las constantes proporcional, derivativa e integral del controlador PID respectivamente y $\triangle u$ es el parámetro de actualización del sensor. A partir de las ecuaciones anteriores se pasa a definir la función que implementa el control PID en software. La función que implementa el PID es la siguiente:

```
def pidController(Kp, Ki, kd,SetPoint,SensorValue, lastError,lastTime, addError):
  """
    deltaU =Kp*(ep + Kd*ed + (1/Ki)*ei)
  """
  sampleTime = 1;

  error = SetPoint - SensorValue
  currentTime = time.time()
  deltaTime = currentTime - lastTime
  deltaError = error - lastError


  ep = error
  ed = (deltaError)/(deltaTime)
  ei = addError + (error*deltaTime)

  output = Kp *(ep +(Kd*ed) + ((1/Ki)*ei))

  datos = [output,error,ei,currentTime]
  print(datos)
  return datos
```

La función recibe como parámetros de entrada las constantes del control PID $K_p$,$K_d$ y $K_i$, el *SetPoint* que es el valor de la señal de referencia, $SensorValue$ que es el valor leído por el sensor, *lastError* el error anterior, *lastTime* que es el instante anterior a realizar la actualización de la variable de control y *addError* que es el error acumulado.

La función devuelve como parámetros de salida un vector que contiene los siguientes datos:

- *output*, es el valor de actualización del sistema.
- *error*, error actual del sistema.
- *ei*, es el error acumulado por el sistema.
- *currentTime*, que es el instante en el que se realiza la actualización del sistema.

La función pidController, a su salida devuelve un parámetro que es el valor de actualización *output*, y a la entrada se tiene el $SensorValue$ y el *SetPoint*, los cuales son valores que vienen en milivoltios y por tanto el valor de actualización de la salida es en milivoltios. En este punto, es claro que el valor de actualización del sensor que se va aplicar es una tensión que servirá para calentar el sensor. Por tanto el valor de actualización debe convertirse a un valor de tensión que haga que el sensor se caliente y sea capaz de llegar a una temperatura que consiga a la salida obtener el valor deseado.

Como se ha visto en capítulos anteriores, el sistema (sensor TGS2600) no tiene definida una relación entre la entrada y la salida que indique cual es la correspondencia entre el valor de la tensión de calentamiento y el valor de tensión de salida. Por tanto, para poder llevar a cabo la conversión entre el valor de salida del controlador PID y la tensión de calentamiento correspondiente se hará una aproximación a partir de una linealización del sistema. Esto se lleva a cabo partiendo del estudio realizado en la sección 4.8, en el cual se ha observado como a medida que se aumenta la tensión de calentamiento del sensor la tensión de salida también lo hace hasta cierto punto en el cual si se la tensión de calentamiento se mantiene fija el valor de la tensión empieza a decaer. Además se ha asumido que se conocen dos puntos de relación entre la tensión de calentamiento aplicada al sensor y la salida.

En la figura 5.5, se representa una señal sinusoidal en función el tiempo que sería la señal de referencia del sistema y en la cual como se puede ver se asume que para el valor máximo de la señal $v_2$ se debe aplicar una tensión de calentamiento $pwm_2$ y para el valor mínimo de la señal $v_1$ se aplica otra tensión de calentamiento $pwm_1$, usando esta pareja de puntos $(pwm_1, v_1)$ y $(pwm_2, v_2)$, se buscará hacer una aproximación lineal para poder encontrar aquel valor de tensión de calentamiento que hace que el sensor sea capaz de seguir la señal en cada instante de tiempo.

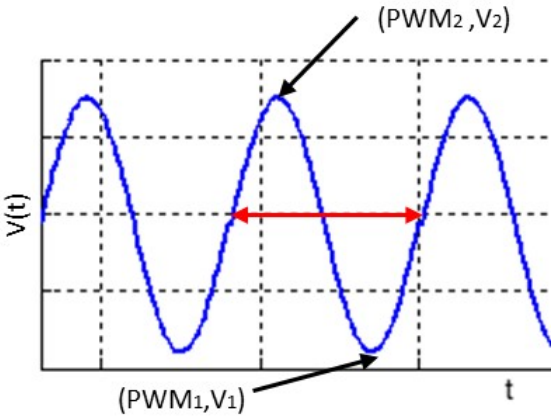



Figura 5.5: Representación de la señal de referencia del sistema en el tiempo, en el que se muestra la pareja de puntos $(pwm_1, v_1)$ y $(pwm_2, v_2)$ en un periodo de la señal.

En la figura 5.6, se representa la relación directa entre la pareja de puntos $(pwm_1, v_1)$ y $(pwm_2, v_2)$ descrita anteriormente, asumiendo que en el eje horizontal se representa la tensión de calentamiento que se aplica al sensor y en eje vertical se representa la salida generada en el sensor TGS2600 [6]. Por tanto, con estos dos puntos conocidos se realizará la linealización, que se describiría con la ecuación 5.6 que es la ecuación de la recta.

$$v_T = \frac{v_2 - v_1}{pwm_2 - pwm_1}(pwm_T - t_1) + v_1 \tag{5.6}$$

Por tanto, en cada periodo de la señal de referencia se tendrá un valor tensión de calentamiento máximo $(pwm_2)$ con el cual se llega a conseguir un valor de tensión máximo $(v_2\ )$ y un valor de tensión mínima $(pwm_1)$ de calentamiento con el que se consigue una tensión mínima $(v_1)$ con estos puntos se busca encontrar aquel valor de tensión de calentamiento $(pwm_T)$ que hace que el sensor consiga una temperatura que refleje a la salida el valor deseado$(v_T)$, ver figura 5.6.

En el eje horizontal en el que se representa la tensión de calentamiento $(PWM)$ en la figura 5.6, se tiene los valores de $PWM_{max}$ y $PWM_{min}$, estos representan los valores máximo mínimo del PWM que se podrán aplicar al sensor y que se han definido previamente en la tabla 5.3. Por tanto, se podrán aplicar tensiones de calentamiento superiores a $(pwm_2)$ o inferiores a $(pwm_1)$, pero siempre habrá unos limite tanto inferior como superior de la tensión de calentamiento que se puede aplicar al sensor.

Una vez definida la forma en la que se hará la conversión entre la salida del controlador y la tensión de calentamiento de actualización, es importante tener en cuenta que el controlador PID devolverá a la salida también el error en ese instante de tiempo. En este punto es importante evaluar el valor del error ya que si este es 0 la actualización de la tensión de calentamiento sera mínima, si por el contrario el error fuese distinto de 0, sería necesario actualizar dicha tensión de calentamiento para conseguir llegar al valor de la señal de referencia.

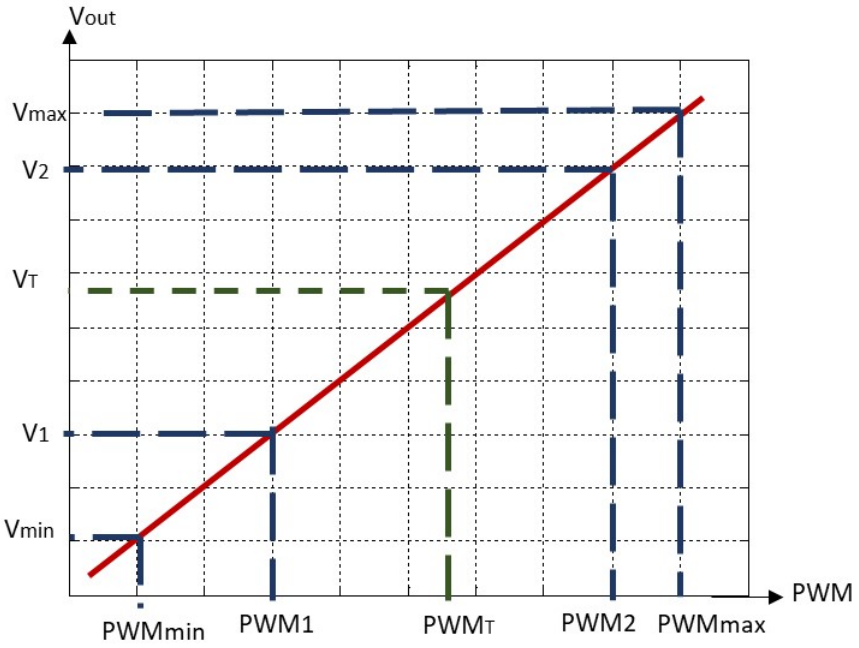



Figura 5.6: Representación de la relación entre los puntos $(pwm_1, v_1)$ y $(pwm_2, v_2)$ para calcular la tensión de calentamiento que se debe aplicar al sensor TGS2600 para poder seguir la señal de referencia.

Cuando el error es distinto de 0, se tendrá que hacer la actualización de la tensión de calentamiento calculando primero la tensión de calentamiento de actualización basándose en el método de linealización que se ha comentado anteriormente. La tensión de calentamiento que se obtenga sera la tensión que se debe aumentar o disminuir a la tensión actual que se esta aplicando al sensor. Además de tener en cuenta el error, es importante tener en cuenta que la tensión actual aplicada el sensor este entre los rangos de tensión de trabajo definidos en las constantes, esto quiere decir que la tensión que se aplique al sensor no puede ser mayor a la constante $temperaturaMax$, ni tampoco puede ser menor a la constante $temperaturaMin$. Por tanto, siempre se va hacer una comprobación que dicha tensión que se vaya aplicar al sensor este entre los rango determinados, si esta tensión estuviese sobrepasando la tensión máxima de trabajo definida la tensión de actualización se caparía a la máxima permitida y si por el contrario la tensión estuviese por debajo de la mínima permitida la tensión aplicada al sensor se caparía a la mínima definida.

**Actualización del target** Tras encontrar el valor la tensión de calentamiento que se debe aplicar al sensor, al final de cada ciclo de seguimiento de la señal de referencia se lleva a cabo el proceso de actualización de la señal. Como se ha mencionado antes, la actualización de la señal de referencia se lleva acabo teniendo en cuenta si los valores del máximo y mínimo leídos en ciclo actual cambian respecto a los valores máximo y mínimo de la señal de referencia. Este proceso de actualización se puede plantear de dos formas:

1. Actualización basada en los valores del máximo y el mínimo. En este caso la señal de referencia se actualiza solo si cambia el máximo o mínimo, esto conlleva que el proceso de exploración del régimen dinámico del sensor bajo la presencia de odorante sea más lento ya que solo tendría en cuanta el cambio de los valores leídos por el sensor y por tanto la actualización de la señal será determinada por el odorante. Este tipo de actualización sería lo más parecido al seguimiento de la señal que se hace en [1].

2. Actualización basada en la tensión de calentamiento aplicada a los valores máximo y mínimo obtenidos por el sensor. Esta actualización no solo tiene en cuenta los valores del máximo y mínimo, sino que además la actualización se realizaría teniendo en cuenta la tensión de calentamiento que se ha aplicado a los valores máximo y mínimo. De esta forma, lo que se busca es poder explorar el régimen del sensor haciendo que este sea capaz de trabajar en todo su rango. Por tanto, con este tipo de actualización se pueden presentar las siguientes situaciones:

   *a*) El sistema consigue seguir la señal, y veo que tengo rango de trabajo sobre la tensión de calentamiento, lo que se hace es intentar ampliar el rango de amplitud de la señal.

   *b*) El sistema no consigue llegar a las condiciones iniciales del ciclo, actualizare la señal teniendo en cuenta los valores del máximo y el mínimo para adaptar la señal y ser capaz de llegar a las nuevas condiciones en el siguiente ciclo.

   *c*) El sistema se sobrepasa el valor del máximo o el mínimo, en el próximo ciclo la señal se ajustara a los nuevos valores del máximo y mínimo, y se podrá forzar esta actualización si sistema tiene rango de trabajo de la tensión de calentamiento para ampliar el rango de amplitud de la señal.

   Con este tipo de actualización se podrá ejercer control sobre la señal de referencia y esta se adaptará dependiendo de condiciones con los valores máximo y mínimo además del rango de tensión de calentamiento que se haya aplicado para llegar a estos valores.

Por tanto, en este trabajo la actualización de la señal de referencia no solo se basa en el cambio de los valores mínimo y máximo de la señal si no que también se basa en los valores de tensión de calentamiento aplicados a dichos puntos de la señal, y como ya se ha mencionado previamente es lo que diferencia la técnica de modulación que se implementa en este trabajo a la que se implementó en [1].

## 5.3. Pruebas de la técnica de modulación

Una vez explicada la técnica de modulación se pasara a mostrar los resultados obtenidos con la implementación de la técnica de modulación en lazo cerrado.

Para poder probar la técnica de modulación implementada en la placas de experimentación se usa el circuito de modulación en amplitud, ya que la técnica busca la temperatura que se debe aplicar al sensor para poder llegar a un valor determinado y por tanto con este circuito se puede realizar la variación de tensión de calentamiento que se debe aplicar al sensor a través de una señal PWM.

Los pasos a seguir para poder ejecutar un experimento de esta modulación serán los mismos que se han explicado en la sección 4.7.1.1, excepto el punto 8, en el que se debe lanzar el siguiente comando para ejecutar el programa que implementa la técnica de modulación.

python3 “nombrePrograma” succión heat2600 tiempo cambio odorante maximoReferencia minimoREferencia

En este caso *nombrePrograma*, sera el nombre del fichero que ejecuta el control del sistema, y los demás parámetros son los parámetros de entrada del sistema explicados en la sección 5.2.1.3.

El primer ejemplo que se mostrara para la técnica de modulación es la primera aproximación que se hace partiendo del proceso explicado en la sección 5.2. En este caso, en este ejemplo se

muestra la técnica de modulación haciendo la actualización de la señal de referencia teniendo en cuenta solo los valores del máximo y mínimo que se leen durante cada periodo. Para este ejemplo se ha ejecutado el programa *principal_v05,2.py*, el código se puede encontrar en el anexo I.5.1.

El vector de conmutación usado para este experimento es el siguiente [1 1 4 3 4], en este caso el tiempo de conmutación de cada electroválvula ha sido de 60 segundos. Como se puede observar en el panel superior de la figurar 5.7, el sensor es capaz de seguir la señal de referencia y se observa como la señal de referencia se va actualizando al final de cada periodo de la señal. En el panel inferior de la figura 5.7, se representa la tensión de calentamiento que se aplica al sensor, que como se puede ver varía a medida que se adapta la señal de referencia.

Aunque en este primer ejemplo el vector de conmutación posee pocas transiciones es fácil darse cuenta que en ningún punto se puede diferenciar en que momento hay un cambio de odorante, ya que la señal de referencia no cambia drásticamente su amplitud debido a que el sensor sigue a esta señal.

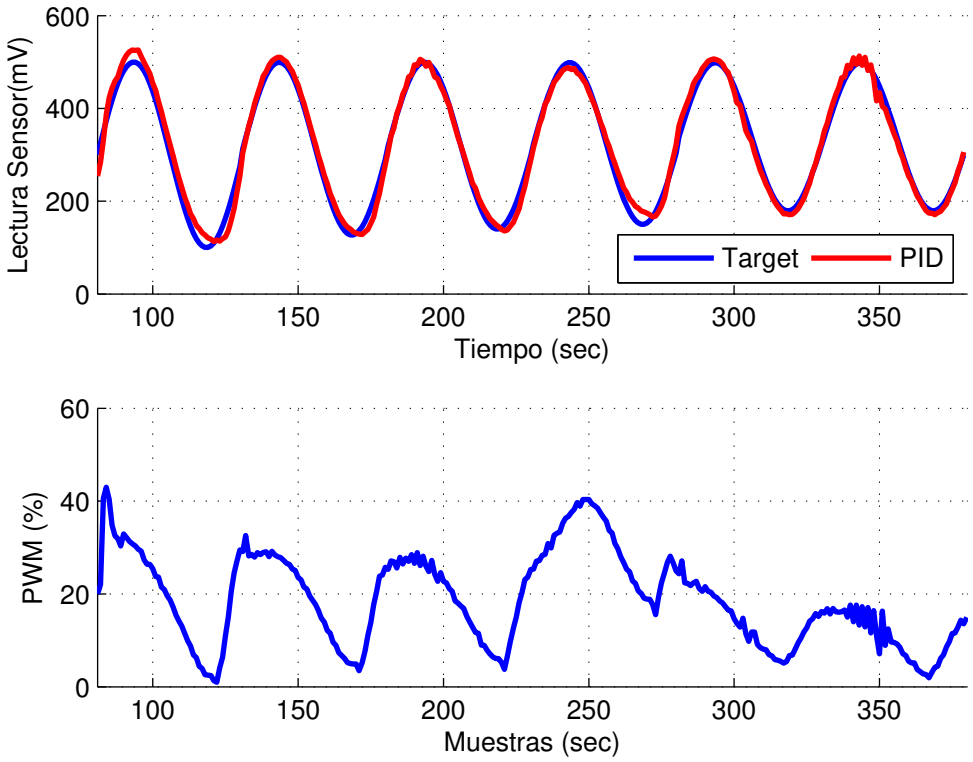



Figura 5.7: Respuesta del sensor al aplicar la técnica de modulación actualizando la señal de referencia basada en los valores del mínimo y el máximo, con un vector de prueba aleatorio en la primera placa de experimentación.

En el siguiente ejemplo que se realiza se usa de nuevo el programa usado en el ejemplo anterior. Para este nuevo experimento se usa el vector de conmutación que contiene todas las transiciones entre los posibles odorantes y el cambio entre cada uno de ellos es de 60 segundos.

En el panel superior de la figura 5.8, se observa el resultado del experimento en el cual se observa como el sensor es capaz de seguir la señal de referencia y además se observa como esta se va actualizando hasta llegar a mantenerse estable. Sin embargo, al igual que ha pasado en el ejemplo anterior resulta complicado distinguir en que instante de tiempo hay una transición entre odorantes. Si se observa el panel inferior de la figura 5.8, en el cual se representa la tensión de calentamiento aplicada al sensor se puede ver los cambios en la amplitud de la señal

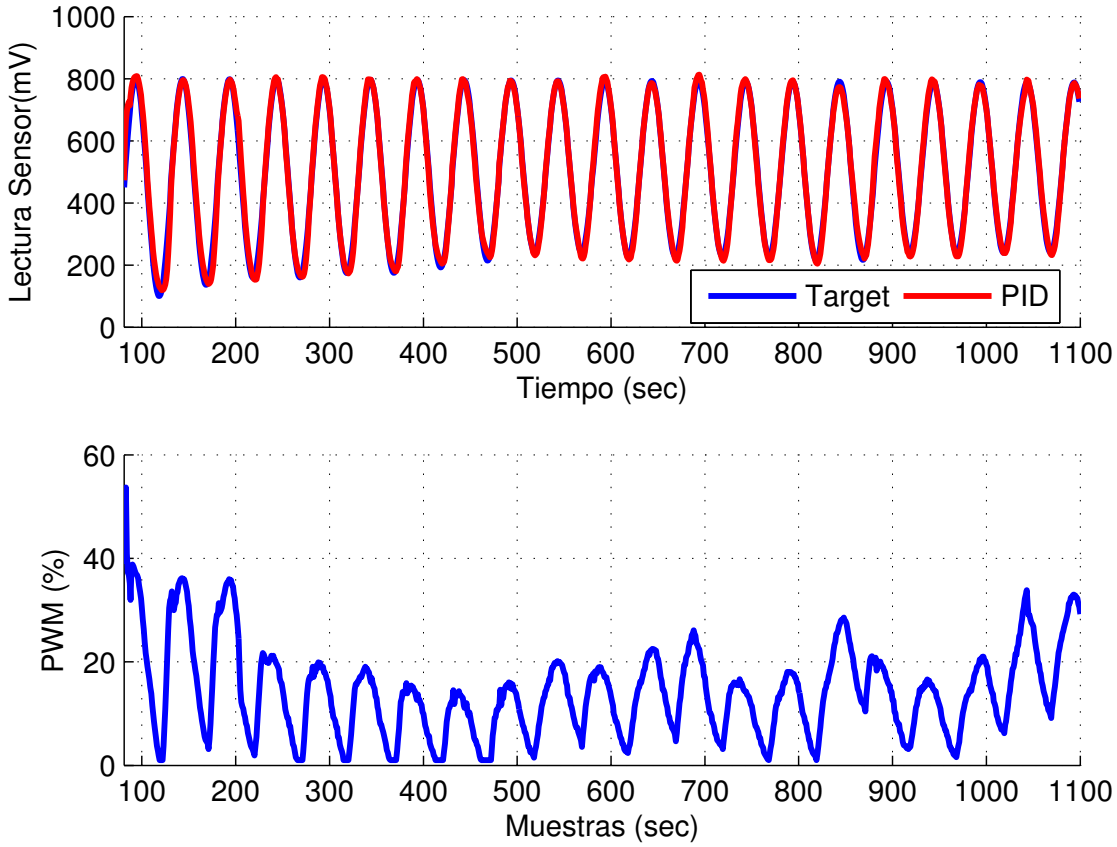



Figura 5.8: Respuesta del sensor al aplicar la técnica de modulación actualizando la señal de referencia basada en los valores del mínimo y el máximo, con un vector de conmutación con todas las transiciones entre odorantes usando la primera placa de experimentación.

de calentamiento con la que se puede ver en que instante de tiempo hay una transición entre odorantes.

Por tanto, se ha observado en las figuras 5.7 y 5.8 que resulta complicado distinguir las transiciones entre odorantes si solo se observa el panel superior de las figuras en las que se representa la lectura del sensor y la señal de referencia. Por otra parte, si observa el panel inferior en el que se representa la tensión de calentamiento del sensor es posible ver los instantes de tiempo en el que se produce la transición entre un odorante y otro.

A continuación se realizaran 3 experimentos con el metanol, etanol y butanol cada uno de forma separada y durante 10 minutos para ver cual es el comportamiento de cada uno de ellos.

Como se puede observar en las figuras 5.9, 5.10 y 5.11 el sensor es capaz de seguir la señal de referencia durante todo el experimento y además se puede ver como la temperatura de calentamiento se mantiene estable, esto quiere decir que la amplitud de la señal de calentamiento es la misma, además de ser también una señal periódica. También se puede observar que con esta primera implementación se deja evolucionar la señal de referencia automáticamente sin tener ningún control sobre ella solamente el seguimiento del mínimo y el máximo en cada ciclo de la señal.

Los ejemplos anteriores, han servido como una primera aproximación de la técnica de modulación en la que el sensor es capaz de seguir la señal. Por tanto, se puede asumir que el controlador PID implementado funciona de forma correcta y además esta es la implementación

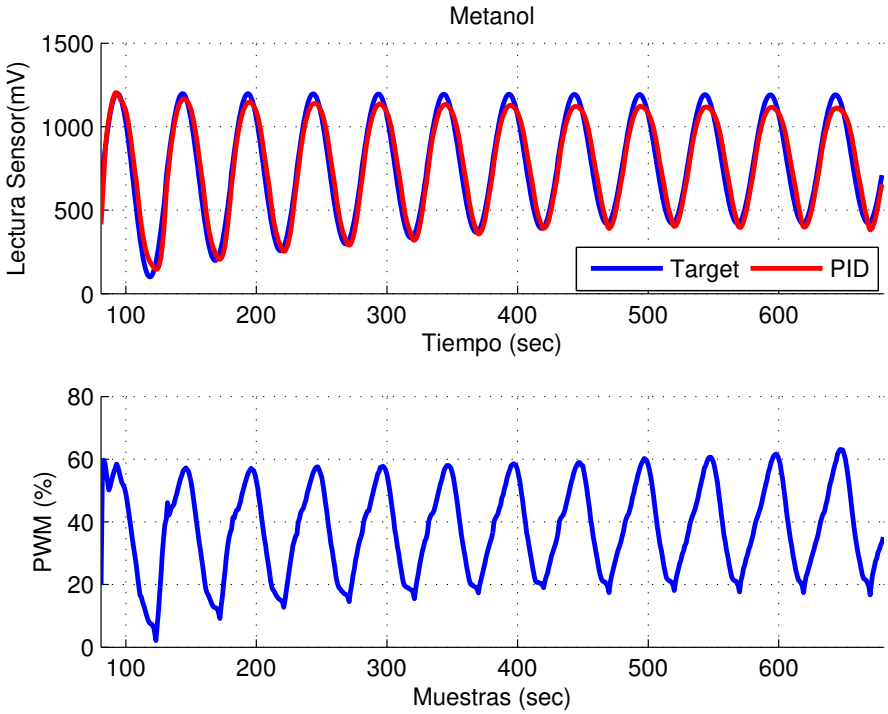



Figura 5.9: Respuesta del sensor al aplicar la técnica de modulación con el metanol durante 10 minutos en la primera placa de experimentación.

mas cercana a la técnica desarrollada en [1]. Además también se ha podido comprobar como el sistema es robusto ya que no se deja que el sensor llegue a un estado estacionario con la presencia de un odorante como se ha mostrado por ejemplo en la figura 5.8, en la que se representan 17 transiciones entre odorantes y se observa como el sistema se adapta cada vez que hay un cambio de odorante cada 60 segundos y este es capaz de seguir la señal.

Por otra parte, se ha observado como el sensor es capaz de seguir la señal de tal forma que no hay una variación de amplitud en la señal de referencia si no que se deja que esta evolucione automáticamente controlando solamente lo valores del máximo y mínimo de la señal de referencia. Sin embargo, el objetivo de la técnica de modulación que sea plantea en este proyecto, no es que el sensor sea capaz de seguir la señal si no que pueda explorar el rango dinámico y sea capaz de modular la señal de referencia bajo la presencia de un odorante, esta es la diferencia principal que hay con respecto ala técnica de modulación que se desarrollo en [1].

Partiendo del programa que se ha ejecutado para los ejemplos anteriores, este se modifica para realizar la actualización de la señal teniendo en cuenta la tensión de calentamiento del sensor de los valores máximo y mínimo de la señal de referencia para así conseguir que el sensor sea capaz de explorar un rango dinámico más amplio cuando esta bajo la presencia de un odorante.

En el siguiente ejemplo se mostrara los resultados de la técnica de modulación tras realizar los debidos cambios al programa anterior. El programa que se debe ejecutar ahora es *principal_v*05,5.*py*, el código se puede encontrar en el anexo I.5.2. En la figura 5.12, se muestra la transición entre el metanol y etanol, en este caso el tiempo de succión de cada odorante se ha aumentado para que así el sensor sea capaz de cambiar su dinámica con el cambio de odorante a lo largo del experimento.

Observando los resultados se ve como la señal de referencia se va modulando en amplitud a lo largo del tiempo, además en este ejemplo se podría determinar en que punto se produce la transición entre un odorante y otro. Por tanto, es indispensable que el tiempo durante el que

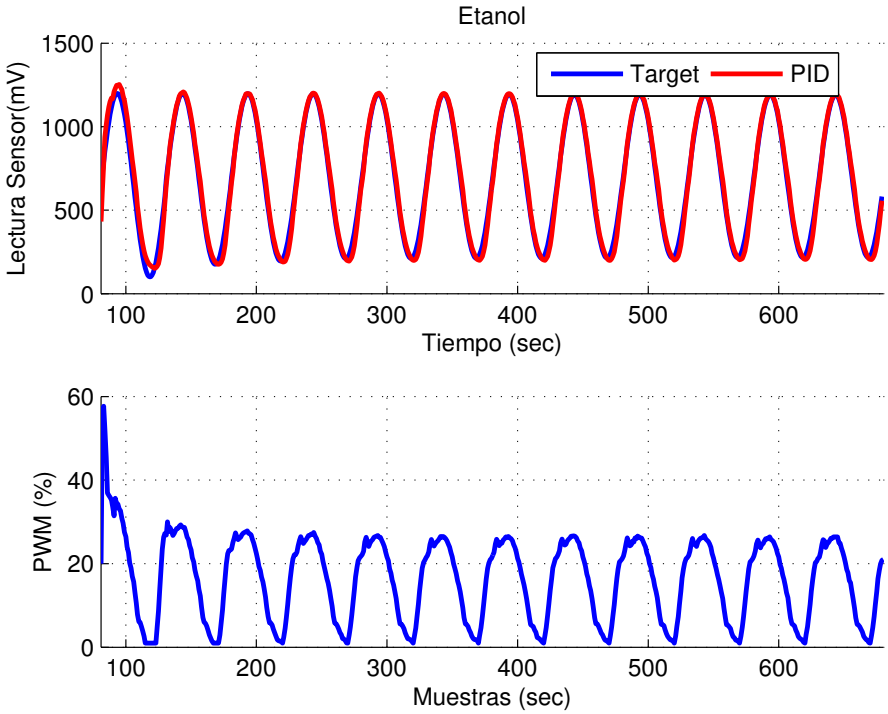



Figura 5.10: Respuesta del sensor al aplicar la técnica de modulación con el etanol durante 10 minutos en la primera placa de experimentación.

se succiona un odorante no sea muy pequeño ya que como se observa la modulación de la señal de referencia no se produce de forma inmediata si no que se debe dar un margen para ver estos cambios. En este caso la técnica de modulación implementada no solo busca ser capaz de seguir la señal si no que además tiene en cuenta la tensión de calentamiento del sensor que se aplicado a los valores máximo y mínimo para así poder forzar al sensor a que pueda explorar la dinámica de cada odorante.

Por último, se comprobara la técnica modulación sobre la segunda placa de experimentación para ello sera necesario ejecutar el programa $principalPID\_placa2\_v1.py$, el codigo se puede encontrar en el anexo I.5.3. En la figura 5.13, se muestra una transición entre el butanol y el metanol en la segunda placa de experimentación. Como se observa durante todo el experimento la señal de referencia es modulada y esta se va adaptando, además se puede ver como la tensión de calentamiento se mantiene estable durante un periodo de tiempo y luego esta empieza a cambiar. Este cambio es debido a la entrada de etanol en el sistema lo que hace que la tensión de calentamiento se adapta para este nuevo odorante.

En los ejemplos anteriores en los que se usado la técnica de modulación implementada, se ha podido observar como el sensor ha sido capaz de explorar los distintos regímenes del odorante consiguiendo la modulación de la señal de referencia. También se ha visto como la tensión de calentamiento es capaz de mantenerse estable y poder seguir la señal de referencia.

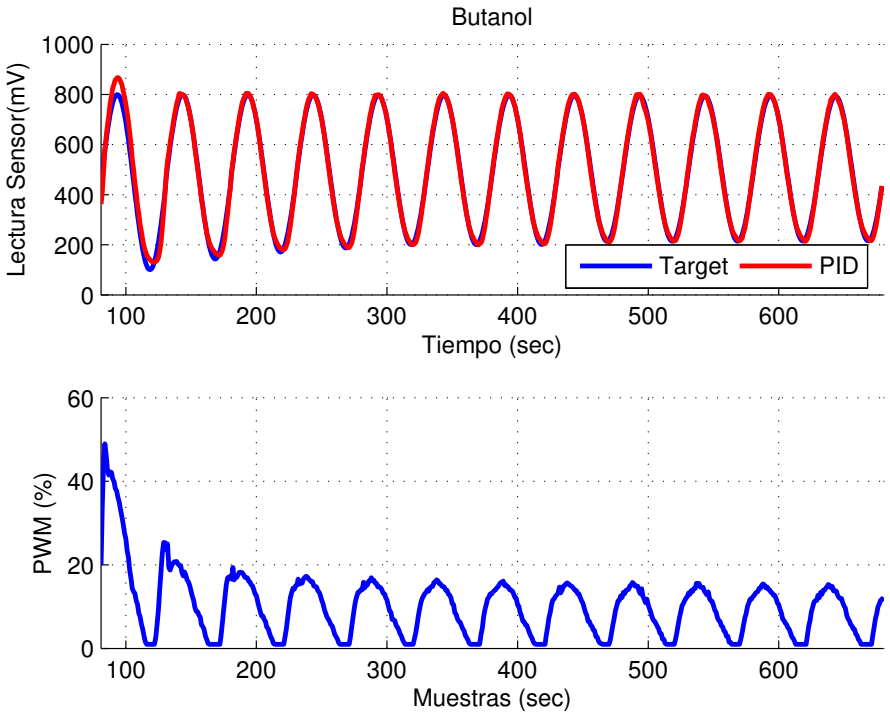



Figura 5.11: Respuesta del sensor al aplicar la técnica de modulación con el butanol durante 10 minutos en la primera placa de experimentación.

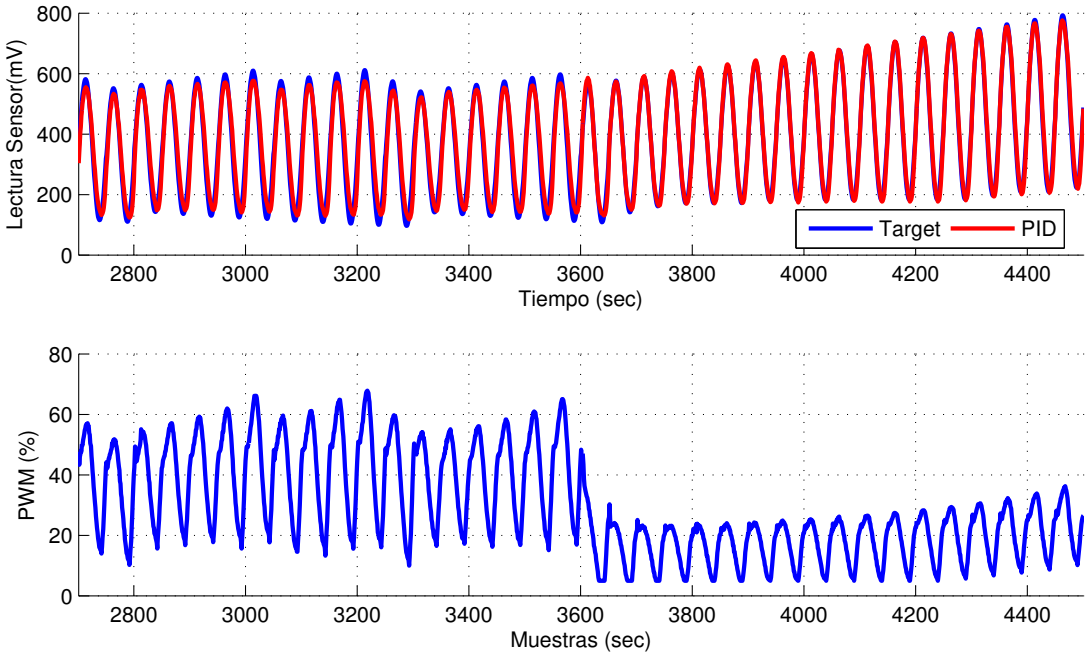



Figura 5.12: Respuesta del sensor al aplicar la técnica de modulación actualizando la señal de referencia basada en los valores del mínimo y el máximo y la tensión de calentamiento aplicada a estos. Representación de la señal obtenida de la transición entre del metanol y etanol.

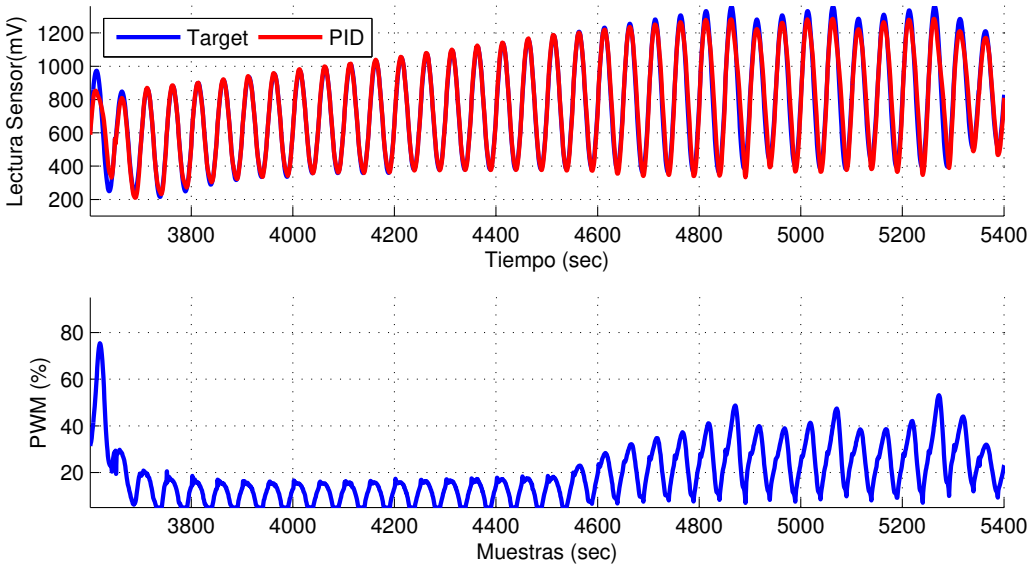



Figura 5.13: Respuesta del sensor al aplicar la técnica de modulación actualizando la señal de referencia basada en los valores del mínimo y el máximo y la tensión de calentamiento aplicada a estos. Representación de la señal obtenida de la transición entre el butanol y etanol.

# 6

# Conclusiones y trabajo futuro

## 6.1. Conclusiones

La realización de este proyecto ha tenido diferentes objetivos que se han basado en la construcción de la plataforma de experimentación y la implementación de la técnica de modulación en lazo cerrado con un controlador PID.

La construcción en PCB de la plataforma de experimentación ha sido uno de los puntos más importantes del proyecto, ya que se parte de un diseño inicial que fue desarrollado en [5], y sobre el cual se busca hacer las mejoras necesarias para que la plataforma sea versátil. Durante el proceso de diseño de la plataforma fue necesario investigar que cosas se podían mejorar respecto al primer diseño, en este caso las mejoras hechas fueron sobre el hardware de la placa inicial, entre ellas el circuito de electroválvulas en el cual se uso una base especial para poder llevar a cabo el proceso de succión sin que hayan perdidas y cumpliendo la funcionalidad deseada.

Al finalizar el proyecto se consigue construir dos plataformas de experimentación. La primera de ellas fue usada para comprobar el funcionamiento de los circuitos y como plataforma de pruebas. Sobre esta placa se comprobó el funcionamiento de las distintas técnicas de modulación que se habían desarrollado previamente. La segunda plataforma construida es una mejora de la primera plataforma y que además cuenta con una electroválvula más que la anterior plataforma. Además de la construcción de la dos plataformas también se construyo un circuito de protección para los puertos ADC de la BBB, ya que esta tiene bastantes limitaciones eléctricas en sus puertos y se podía estropear fácilmente. Cabe resaltar que durante todo el proyecto no se quemo ninguna placa BBB gracias al uso del circuito de protección usado para los puertos ADC y a los circuitos que ya se habían diseñado para la protección de pines de la placa.

En paralelo a la construcción de las plataforma también se estuvo trabajando en la implementación de la técnica de modulación en lazo cerrado con un controlador PID. La técnica implementada ha supuesto un estudio sobre los controladores PID y también el comportamiento del sensor TGS2600. La técnica que se ha implementado busca explorar el régimen del sensor modificando dinámicamente el máximo y el mínimo de la señal cuando esta expuesto a un odorante, por tanto está es la diferencia principal y lo que la hace diferente de la técnica desarrollada en [1].

El desarrollo de este trabajo a supuesto un trabajo multidisciplinar, ya que no solo se ha

trabajado en la construcción de una plataforma o la implementación de una técnica de modulación, si no que también se han realizado tareas como aprender hacer las muestras de los odorantes para los diferentes experimentos, uso de elementos de laboratorio, además del trabajo en equipo ya que se ha usado el software desarrollado en [33] para integrarlo en las plataformas que se han construido (exceptuando la función que realiza el controlador PID, que es la que se ha implementado como una de las partes originales de este proyecto) y verificar su correcto funcionamiento.

## 6.2. Trabajo futuro

Este proyecto ha conseguido integrar sobre una misma plataforma de experimentación distintas técnicas de modulación, lo que hace que se planteen nuevos retos sobre este trabajo. Durante el desarrollo del proyecto se han podido ver distintas cosas que se pueden mejorar tanto a nivel hardware como a nivel software.

A nivel hardware siempre se podrá mejorar la plataforma de experimentación y adecuarla a nuevas técnicas de modulación que se quieran implementar. Una mejora que se propone en las plataformas de experimentación construidas, es respecto al sistema de electroválvulas el cual cumple la funcionalidad que se busca para el sistema pero presenta la desventaja de tener un consumo de potencia alto debido a la base de montaje que se usa, ya que para dejar el paso de un odorante en cada instante de tiempo, las electroválvulas que están conectadas a los odorantes que no se quiere dejar pasar se les debe aplicar energía para bloquear el paso de estos odorantes.

Respecto a la técnica de modulación que se ha implementado, actualmente esta es capaz de seguir una señal sinusoidal. Una mejora que se propone es adaptar la técnica implementada para que esta sea capaz de seguir distintas señales de referencia como una señal de onda cuadrada, una señal triangular, etc. También se propone llevar a cabo una mejora del modelo lineal para el cálculo de la tensión de calentamiento del sensor que predice la tensión de salida.

Por otra parte, se propone usar la técnica de modulación implementada en este proyecto para la clasificación de odorantes y comprobar si hay una mejora respecto a lo que hay actualmente [1].

# Bibliografía


[1] Fernando Herrero-Carrón, David Yañez, Francisco de Borja Rodríguez, and Pablo Varona. An active, inverse temperature modulation strategy for single odorant classification. *Sensors Actuators B: Chemical*, 206:555–563, 2015.

[2] David Yañez. Estrategias bioinspiradas para la adquisición de olores en narices. Trabajo de fin de máster, Universidad Autónoma de Madrid, 2009.

[3] Miguel Macías, Antonio García, Carlos Javier García, Horacio Manuel González, Ramón Gallardo, and Juan Carlos Peguero. Acetic acid detection threshold in synthetic wine samples of a portable electronic nose. *Sensors*, (13):208–220, 2013.

[4] Alejandro Pequño Zurro. Uso de una nariz electrónica ultra-portátil en robots para la detección de fuentes odorantes. Proyecto fin de carrera, Universidad Autónoma de Madrid, 2015.

[5] Sergio De La cruz Gutiérrez. Desarrollo de una plataforma para discriminación de odorantes mediante técnicas de modulación dinámica. Proyecto fin de carrera, Universidad Autónoma de Madrid, 2015.

[6] Fígaro. *Fígaro Gas Sensor TGS260*. http://www.figarosensor.com.

[7] David Yañez, Adolfo Toledano, Eduardo Serrano, Ana Martí de los Rosales, Francisco de Borja Rodríguez, and Pablo Varona. Characterization of a clinical olfactory test with an artificial nose. *Frontiers in Neuroengineering*, 5(13):1–5, February 2012.

[8] Carlos García-Saura, Francisco de Borja Rodríguez, and Pablo Varona. Design principles for cooperative robots with uncertainty-aware and resource-wise adaptive behavior. In Springer, editor, *Biomimetic and Biohybrid Systems*, pages 108–117, 2014.

[9] Tomás Vasquez Rubio. Integración de una nariz electrónica ultra-portatil en un robot modular para el control de su movimiento a través de los odorantes recibidos. Proyecto fin de carrera, Universidad Autónoma de Madrid, 2012.

[10] Carlos García-Saura. Cooperative strategies for the detection and localization of odorants with robots and artificial noses. Trabajo de fin de grado, Universidad Autónoma de Madrid, 2014.

[11] Alexander Vergara, Eduard Llobet, Jesús Brezmes, Xavier Vilanova, Peter Ivanov, Isabel Gràcia, Carles Cané, and Xavier Correig. Optimized temperature modulation of micro-hotplate gas sensors through pseudorandom binary sequences. *IEEE SENSORS JOURNAL*, 5(6):1369–1378, December 2005.

[12] Alexander Vergara, Eduard Llobet, Jesús Brezmes, Xavier Vilanova, Peter Ivanov, Isabel Grácia, and Carles Cané andXavier Correig. Quantitative gas mixture analysis using temperature-modulated micro-hotplate gas sensors: Selection and validation of the optimal modulating frequencies. *Sensors Actuators B: Chemical*, 123:1002–1016, 2007.

[13] Eugenio Martinelli, Davide Polese, Alexandro Catini andArnarldo D'Amico, and Corrado Di Natale. Self-adapted temperature modulation in metal-oxide semiconductor gas sensors. *Sensors Actuators B: Chemical*, 16:534–541, 2012.

[14] Andrew P. Lee and Brian J. Reedy. Temperature modulation in semiconductor gas sensing. *Sensors Actuators B: Chemical*, 60:35–42, 1999.

[15] Derek Molloy. *Exploring BeagleBone*. John Wiley & Sons, Inc, 2015.

[16] Katsuhiko Ogata. *Ingeniería de control moderna*. PEARSON, 2010.

[17] T. C. Pearce, S. S. Schiffman, H.T. Nagle, and J.W. Gardner. *Handbook of machine olfaction: electronic nose technology*. WILEY-VCH Verlag GmbH & Co. KGaA,Weinheim, 2003.

[18] *TravTigerEE*. https://commons.wikimedia.org/wiki/File:PID_en_updated_feedback.svg.

[19] *Controladores PID*. http://www.dia.uned.es/ fmorilla/MaterialDidactico/El%20controlador%20PID.pdf.

[20] J.G. Ziegler and N.B. Nichols. Optimumsettings for automatic controllers. pages 759–765, 1942.

[21] Matt Richardson. *Getting Started With BeagleBone*. Maker Media Inc, 2014.

[22] Mark A. Yoder & Jason Kridner. *BeagleBone Cookbook*. Maker Media Inc, 2015.

[23] SCM. *Series SY100*. http://docs-europe.electrocomponents.com/webdocs/144e/0900766b8144e4fc.pdf.

[24] RS. *RS D220BL Micro Pump*. http://docs-europe.electrocomponents. com/webdocs/1313/0900766b81313b69.pdf.

[25] *Digital-output relative humidity & temperature sensor/module DHT22*. https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf.

[26] *xx555 Precision Timers*. http://www.ti.com/lit/ds/symlink/ne555.pdf.

[27] *ULN200x, ULQ200x High-Voltage, High-Current Darlington Transistor Arrays*. http://www.ti.com/lit/ds/symlink/uln2003a.pdf.

[28] *TGS 2611 - for the detection of Methane*. http://www.figarosensor.com/products/2611pdf.pdf.

[29] *PN2222, PN2222A. General Purpose Transistors*. https://www.onsemi.com/pub/Collateral/PN2222-D.PDF.

[30] *TIP120, TIP121, TIP122 (NPN); TIP125, TIP126, TIP127 (PNP). Plastic Medium-Power Complementary Silicon Transistors*. https://www.onsemi.com/pub/Collateral/TIP120-D.PDF.

[31] *1N4001*. https://www.diodes.com/assets/Datasheets/ds28002.pdf.

[32] *R-78E-0.5*. https://www.recom-power.com/pdf/Innoline/R-78Exx-0.5.pdf.

[33] Angel Fuente Ortega. Estudio y análisis de diferentes técnicas de modulación en la adquisición de un odorante en el contexto de narices electrónicas. Trabajo de fin de máster, Universidad Autónoma de Madrid, 2018.

[34] *MCP6001/1R/1U/2/4*. http://ww1.microchip.com/downloads/en/DeviceDoc/21733j.pdf.

# A

# Presupuesto

A continuación, en la tabla A.1 se presenta un presupuesto en el que se refleja los materiales usados para la construcción de las placas de experimentación.

| **Cantidad** | **Articulo** | **Precio unidad** | **Precio Total** |
|---|---|---|---|
| 25 | ULN2003AN | 0,24 € | 6,00 € |
| 10 | Transistor PN2222ATA | 0,14 € | 1,36 € |
| 10 | Zócalo 8 contactos | 0,57 € | 5,68 € |
| 10 | Zócalo 16 contactos | 0,28 € | 2,82 € |
| 10 | TIP120G | 0,41 € | 4,09 € |
| 20 | Resistencias | 0,06 € | 1,16 € |
| 15 | Condensador 100uF | 0,20 € | 3,00 € |
| 10 | AO MCP6002 | 0,29 € | 2,92 € |
| 10 | AO MCP6004 | 0,46 € | 4,60 € |
| 10 | Zócalo 14 contactos | 0,26 € | 2,61 € |
| 1 | Fuente alimentación 24 V | 11,89 € | 11,89 € |
| 5 | Conector DC hembra 5 mm | 0,90 € | 4,50 € |
| 1 | Cable alimentación | 4,23 € | 4,23 € |
| 1 | Fuente alimentación 24 V | 17,27 € | 17,27 € |
| 11 | Electroválvulas | 19,14 € | 210,54 € |
| 10 | Cable electroválvulas | 4,57 € | 45,70 € |
| 2 | Motores | 56,55 € | 113,10 € |
| 2 | Base montaje electroválvulas | 43,41 € | 86,82 € |
| 1 | Sensor TGS2600 | 11,00 € | 11,00 € |
| 1 | Sensor temperatura y humedad | 13,00 € | 13,00 € |
| 2 | Placas BBB | 69,95 € | 139,90 € |
| | varios | 20,00 € | 20,00 € |
| | | **Total** | 693,05 € |

Tabla A.1: Presupuesto de la construcción de las plataformas de experimentación.

# B

# Configuración placa BBB

## B.1. Configuración y gestión de usuario

### B.1.1. Primera conexión

Lo primero que se debe hacer es conectar la placa a través del cable USB que se proporciona. A continuación se debe abrir una terminal e introducir el siguiente comando:

*ssh root@192.168.72*

Tras la primera conexión la placa pedirá que se introduzca una contraseña, por defecto la placa viene configurada sin contraseña por lo cual con pulsar la tecla enter se conectara y se estará con el usuario de root.Una vez dentro, se debe cambiar la contraseña al usuario para ello por consola se introduce el comando:

*passwd*

Se pediré que se introduzca la contraseña dos veces y así está ya estará cambiada. Es importante que la contraseña sea segura, que contenga una combinación de letras mayúsculas, minúsculas, signos, numeros. Se recomienda que la contraseña sea fácil de recordar, esto no quiere decir que esta sea demasiado corta.

### B.1.2. Creación usuario

Después de poder conectarse a la placa es importante poder crear un nuevo usuario, así como la respectiva contraseña, o en su defecto usar el usuario Debian que tiene la placa. Para la creación de un nuevo usuario se debe introducir en la consola el siguiente comando:

*adduser newUser*

Se pedirá inmediatamente una contraseña para el usuario que se ha creado, una vez introducida la contraseña se pedirá que se introduzca de nuevo la contraseña para comprobar que las contraseñas coinciden. La nueva contraseña debe seguir las recomendaciones mencionadas anteriormente. De esta forma ya se tendría creado el nuevo usuario.

Por otra parte, se puede hacer que el nuevo usuario pueda tener permisos de super usuario. Para ello es necesario añadirlo al grupo supo, eso se hace introduciendo el siguiente comando:

*adduser user sudo*

*usermod -a -G sudo user*

Es importante que para llevar a cabo esta acción se deba estar previamente dentro del sistema como usuario sudo, además que después de esta acción haya que reinicia la placa para que se efectúen los cambios realizados.

## B.2. Configuración de red y DNS

La placa BBB al contar con una conexión ethernet hace que se pueda gestionar el acceso a la red y por tanto facilita el poder trabajar en remoto. Por tanto, se pasará a explicar cómo se debe llevar a cabo la configuración de la red y DNS.

### B.2.1. Configuracion de la interfaz

Para llevar a cabo la configuración de la interfaz se debe editar el fichero de interfaces que es el encargado de la gestión de la interfaz de red de la placa. Para realizar los cambios sobre este fichero se debe estar dentro del sistema como super usuario. En consola se debe escribir el siguiente comando:

*sudo vi /etc/network/interfaces*

Una vez abierto el fichero, se debe ir hasta el final del mismo para introducir las siguientes líneas:

- *auto eth0*
- *iface eth0 inet static*
- *address* $IP_{asignada}$
- *netmask* $IP_{mascara}$
- *gateway* $IP_{gateaway}$

Se deben guardar los cambios realizados y salir del editor. Los datos de $IP_{asignada}$, $IP_{mascara}$, $IP_{gateaway}$ son datos que nos debe proporcionar la universidad, esta gestión se debe llevar a cabo con el tutor de TFM y lo único que se debe aportar él es número de roseta donde se quiere realizar la conexión ethernet. Estos datos se envían a través de correo electrónico.

A continuación, se debe llevar a cabo la configuración del DNS la cual se realizara dependiendo del SO que tenga la BBB. Lo primero que se debe hacer ir a al siguiente archivo, introduciendo por consola el comando:

*vi /etc/ resolv.conf*

Si al abrir el fichero hay un mensaje que dice "No modificar el contenido porque se borrará", no se debe cambiar nada.La configuración del DNS se hará en el siguiente archivo, para ello se introduce el siguiente comando:

*sudo vi /etc/network/interfaces*

Al final del archvio se introduce la siguiente línea:

*dns-nameservers* $IP_{DNS1}$, $IP_{DNS2}$

Si no es así se podrá modificar el fichero borrando todo lo que hay dentro y se debe poner lo siguiente:

*nameserver* $IP_{DNS1}$

*nameserver* $IP_{DNS2}$

Se sale y se guardan los cambios, el fichero quedará como se muestra en la la figura B.1

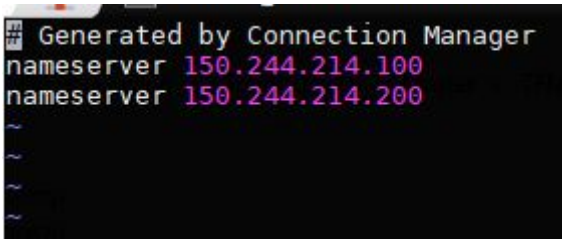



Figura B.1: Archivo de configuración DNS

### B.2.2. Cambiar nombre al host

Para llevar a cabo esta acción por el nombre del host que ha proporcionado la universidad, se deben realizar cambios sobre dos archivos. Primero abriremos el siguiente archivo a través del siguiente comando:

*sudo vi /etc/hosts*

De este archivo como se observa en la figura B.2, las dos primeras líneas empiezan por una IP. A continuación, viene la información acerca de cómo el sistema está configurado en diferentes hosts. En el caso de nuestra placa nos interesa las palabras de la segunda línea en el cual está el nombre del host que se nos asignado, en este caso "maquinaNariz".

El segundo fichero que se debe abrir, introducimos en consola el siguiente comando:

*sudo vi /etc/hostname*

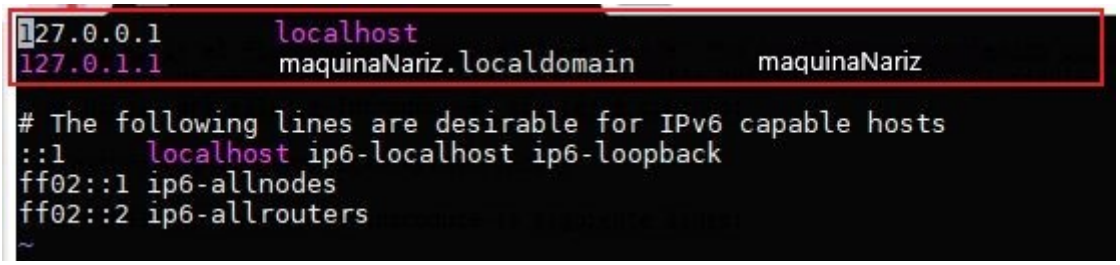



Figura B.2: Archivo de configuración para cambiar nombre del hosts


Figura B.3: Archivo de configuración de hostname

De este fichero se borra todo el contenido y se debe poner el nombre que se haya asignado a la BBB, como se muestra en la figura B.3

Por último, para que los cambios sean efectivos se puede reiniciar la BBB con el siguiente comando:

**sudo reboot**

O reinicar la interfaz de red con el siguiente comando:

*sudo service netwroking restart*

### B.2.3. Ajustar la hora del SO

La hora de la placa viene configurada por defecto con el huso horario de Londres, por tanto para poder cambiar al huso horario que quiera se debe hacer lo siguiente a través de la consola:

*cd /usr/share/zoneinfo/Europe*

Se lista todas las ciudades que hay en el directorio, con ayuda del comando *ls*. Se selecciona la ciudad de que se quiere poner la hora, en este caso Madrid. A continuación, se debe borrar el siguiente fichero, con el siguiente comando:

*sudo rm /etc/localtime*

Este fichero es el encargado de configurar la hora del sistema. Cuando el fichero se ha borrado se deberá crear un enlace simbólico a la ciudad que se haya seleccionado. Para crear este enlace se introduce el siguiente comando:

*ln -s /usr/share/zoneinfo/Europe/Madrid /etc/localtime*

Después de hacer esto se debe reiniciar la placa y se tendría configurada el SO con el huso horario de Madrid

## B.3. Instalación de paquetes fundamentales para Python

En este apartado se explicará cómo instalar los paquetes de Python para poder trabajar con la BBB. Lo primero que se hará es actualizar los paquetes que se encuentran instalados en la BBB, para ello se lanza el siguiente comando por consola:

*sudo apt-get update*

Una vez finalizada la actualización se pasará a la instalación de las diferentes librerias que se van a usar.

### B.3.1. Adafrui_BBIO

Esta es una librería libre la cual ha sido desarrollado por Adafruit para poder llevar a cabo el control d elos pines de BBB con Python. Por tanto, primero se deben descargar las dependencias introduciendo el siguiente comando:

*sudo apt-get install build-essential python-dev python-setuptools python-pip python-smbus -y*

Después se instalará la librería con el siguiente comando:

*sudo pip install Adafruit_BBIO*

### B.3.2. Adafrui_DHT

Esta librería también es de código libre, y se encarga de comunicarse con el sensor de temperatura y humedad para así poder obtener los valores correspondientes. El codigo de esta librería se descarga desde Github, se introduce el siguiente comando para su descarga:

*git clone https://github.com/adafruit/Adafruit_Python_DHT.git*

*cd Adafruit_Python_DHT*

*sudo python setup.p install*

### B.3.3. SciPy

Esta también es una librería de código libre la cual contiene herramientas y algoritmos matemáticos para Python. La instalación se realiza con el siguiente comando:

*sudo apt-get install python-scipy*

### B.3.4. NumPy

Liberia libre de Python, la cual tiene funciones para trabajar con vectores y matrices. La instalación se realiza con el siguiente comando:

*sudo apt-get install python-numpy*

## B.4. Apagado BBB

En este apartado se explicará cómo se debe apagar correctamente la BBB, ya que si no hace de la forma correcta la placa puede estropearse o hacer que el tiempo de arranque del SO aumente. Las pautas para apagar, reiniciar y arrancar la BBB son las siguientes:

- Para apagar la placa se escribirá por terminal siguiente comando *shutdown -h now*, esto hará que la placa se apague cerrando todos los procesos o programas abiertos.
- Para reiniciar la placa se escribirá por terminal el comando *reboot*, reiniciando la placa de forma correcta.
- Pulsar el botón de apagado/encendido una vez para que la placa se paga de forma correcta.
- Mantener el botón de apagado/encendido alrededor de unos 7 segundos, esto hará que se lleva a cabo un apagado forzado de la placa. Esto se debe evitar hacer excepto porque la placa se encuentre bloqueada y no se posible realizar su apagado con las opciones anteriores.
- Cuando la placa está apagada, con el conector de alimentación o el cable USB se puede presionar el botón de apagado/encendido para arrancar la placa.
- No arrancar la tarjeta conectado y desconectando la alimentación, mejor usar el botón de encendido así la duración de la placa será mayor.

# C

# Esquemático y huellas del circuito de protección para puertos ADC de la BBB

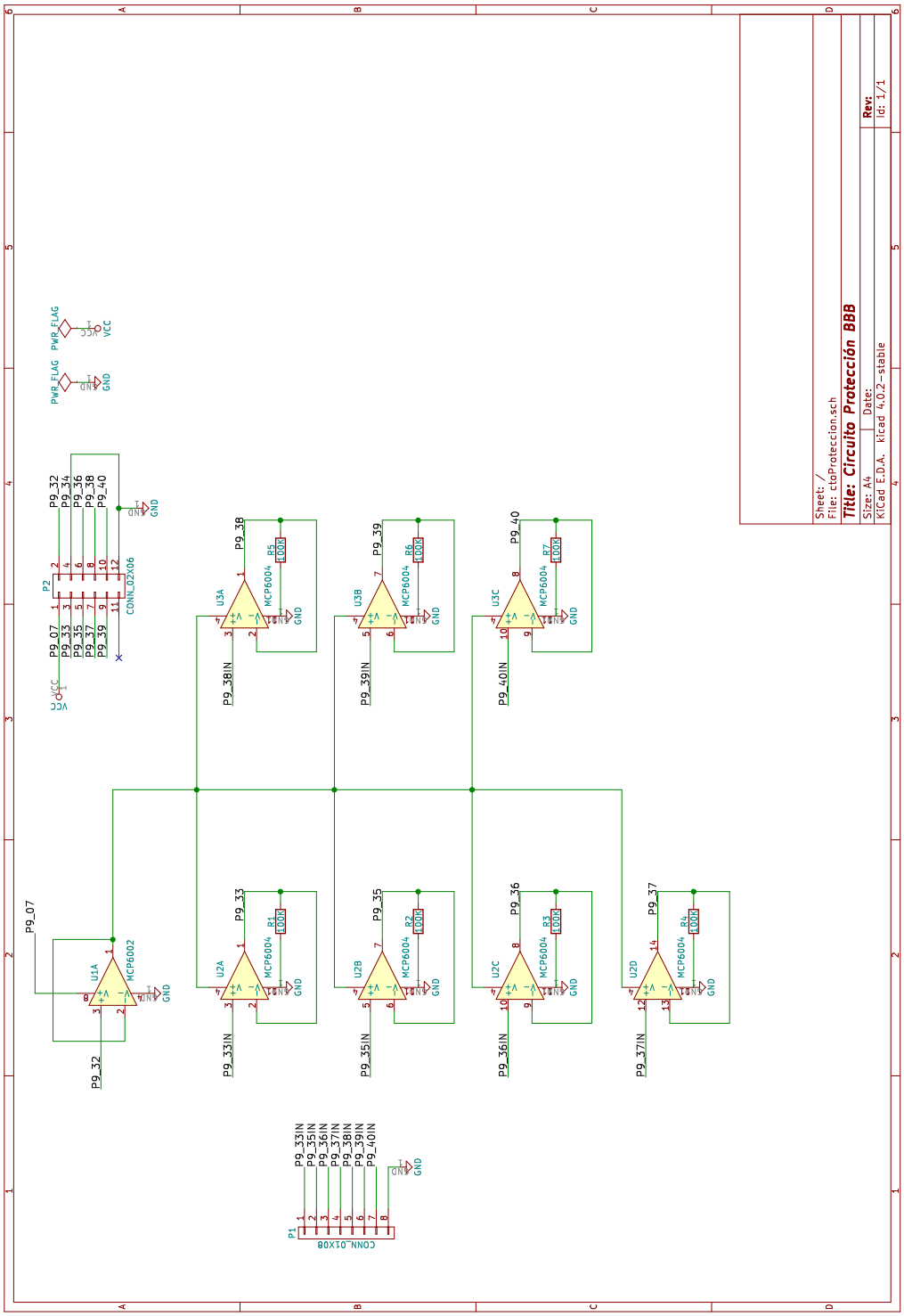



Figura C.1: Esquemático del circuito de protección para puerto ADC de la BBB.

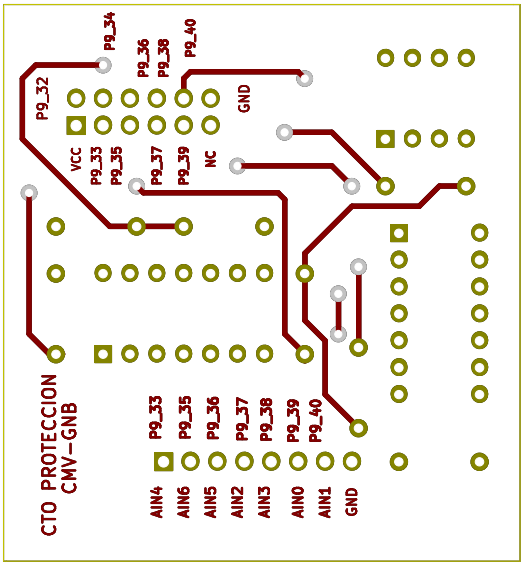



Figura C.2: Capa top del circuito de circuito de protección.

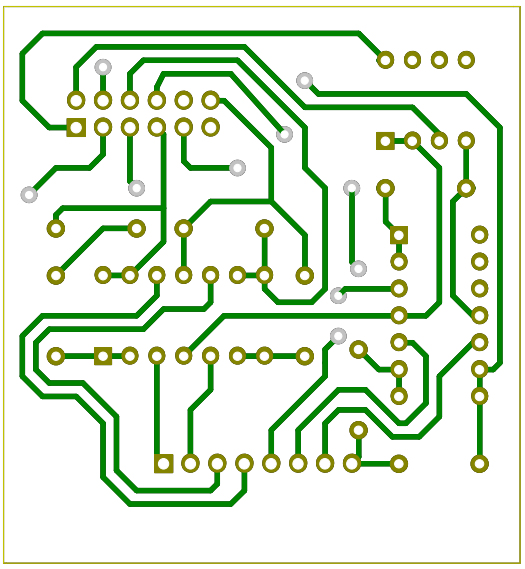


Figura C.3: Capa bottom del circuito de circuito de protección.

# D

# Esquemático y huellas de la primera plataforma de nariz electrónica

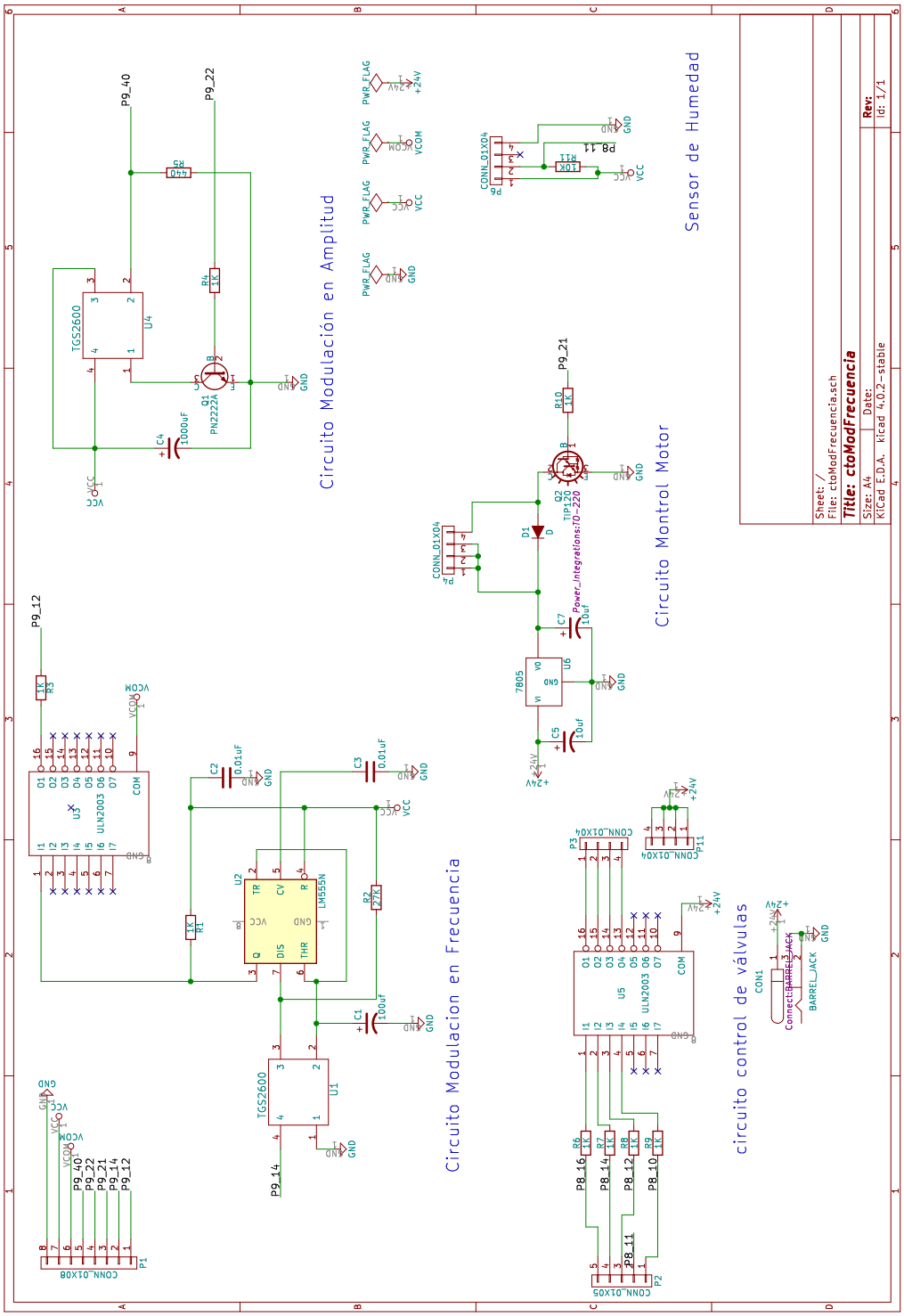



Figura D.1: Esquemático de la primera placa de experimentación.

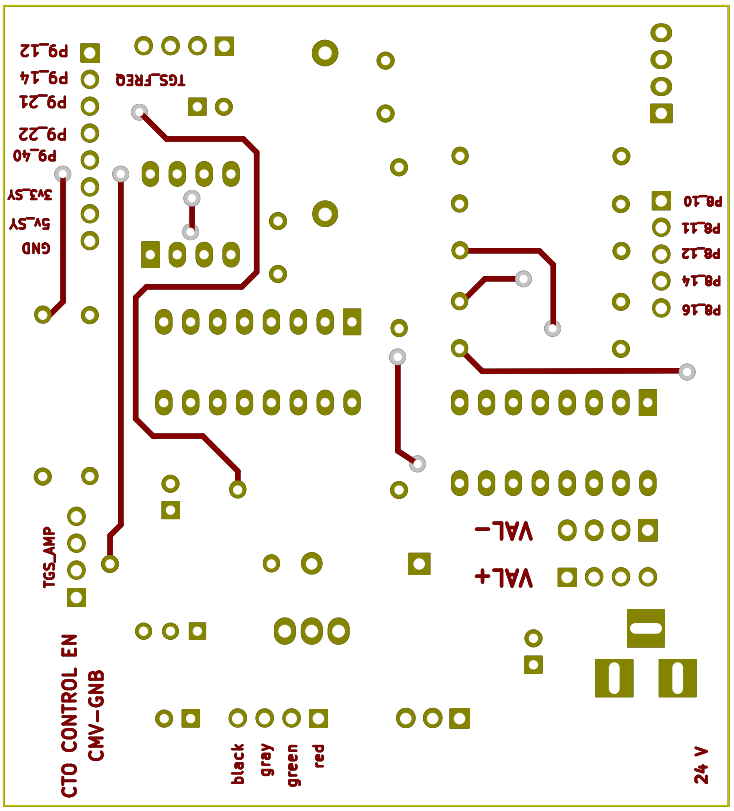



Figura D.2: Capa top del circuito de la primera placa de experimentación.

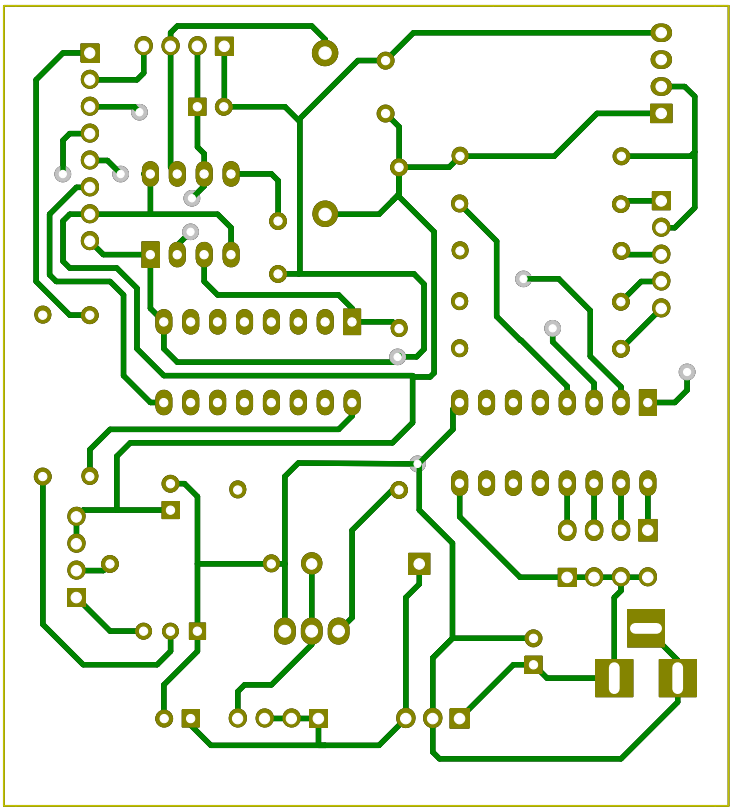


Figura D.3: Capa bottom del circuito de la primera placa de experimentación.

# E

# Esquemático y huellas de la segunda plataforma de nariz electrónica

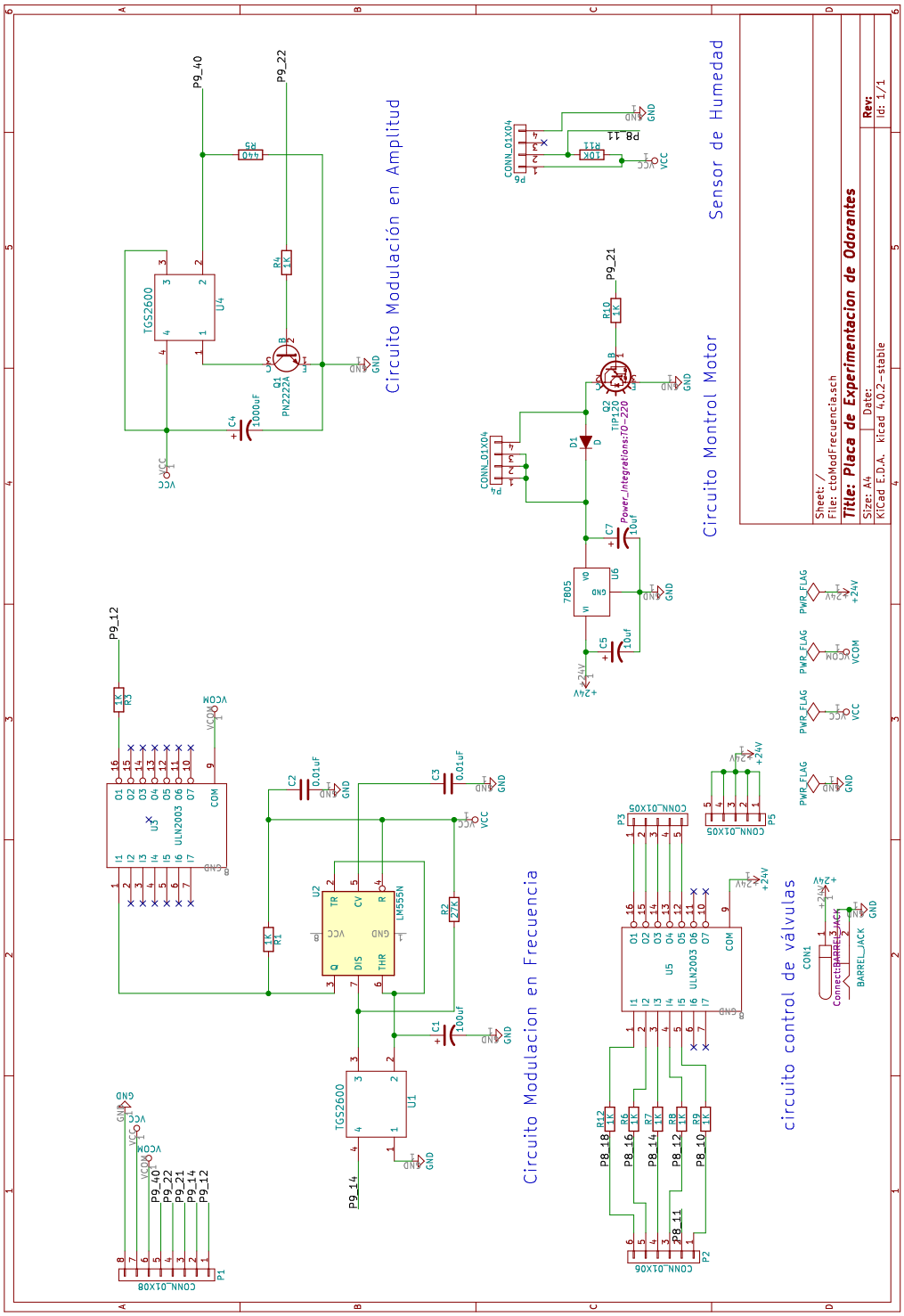



Figura E.1: Esquemático de la segunda placa de experimentación.

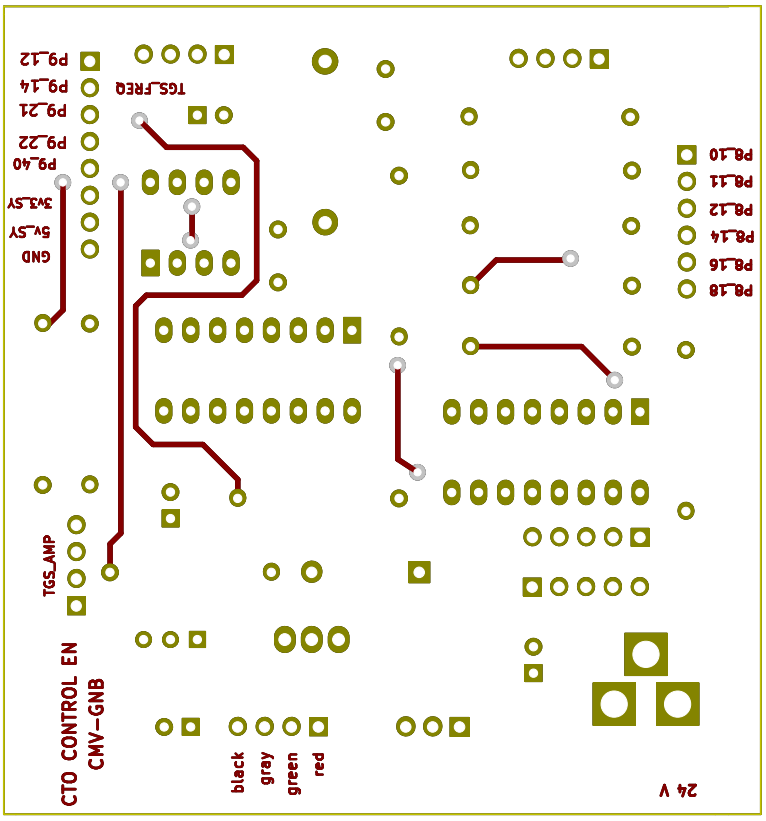



Figura E.2: Capa top del circuito de la segunda placa de experimentación.

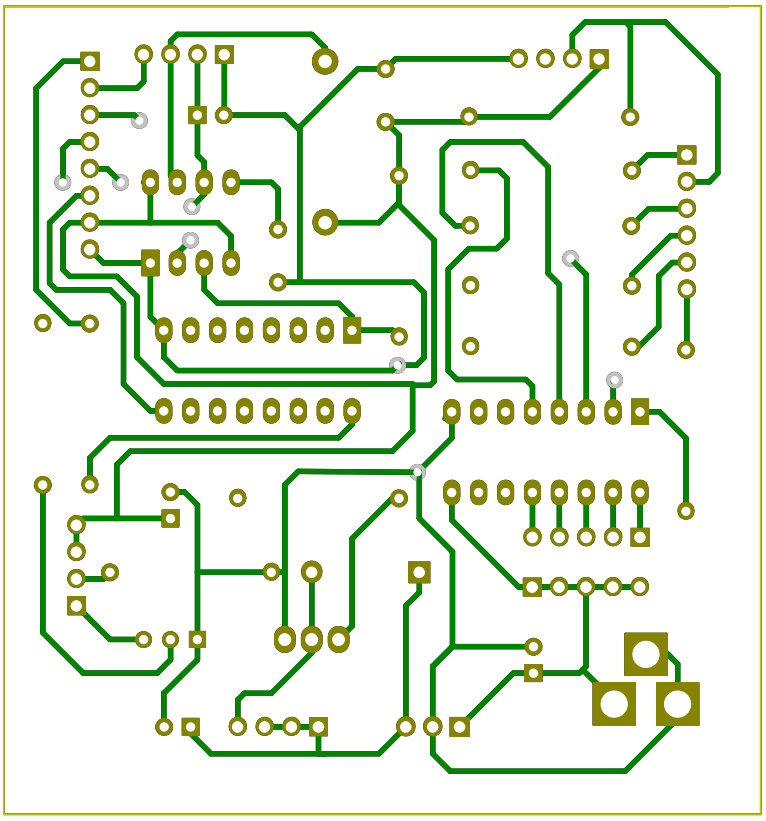


Figura E.3: Capa bottom del circuito de la segunda placa de experimentación.

# F

# Diagrama de pines BBB

En la figura F.1, se muestran la configuración por defecto que tiene la placa BBB. Esta placa esta formado por 92 pines cada uno de ellos con diferentes funciones que pueden ser configuradas [15].

**P9**

| Función | Pines físicos | | Función |
|---|---|---|---|
| DGND | 1 | 2 | DGND |
| VDD 3,3 v | 3 | 4 | VDD 3,3 v |
| VDD 5V | 5 | 6 | VDD 5V |
| SYS 5V | 7 | 8 | SYS 5V |
| PWR_BUT | 9 | 10 | SYS RESET |
| UART4_RXD | 11 | 12 | GPIO_60 |
| UART4_TXD | 13 | 14 | EHRPWM1A |
| GPIO_48 | 15 | 16 | EHRPWM1B |
| SPIO_CSO | 17 | 18 | SPIO_D1 |
| I2C_SCL | 19 | 20 | I2C_SDA |
| SPIO_DO | 21 | 22 | SPIO_SCLK |
| GPIO_49 | 23 | 24 | UART1_TXD |
| GPIO_117 | 25 | 26 | UART1_RXD |
| GPIO_115 | 27 | 28 | SP11_CSO |
| SP11_DO | 29 | 30 | GPIO_112 |
| SP11_SCLK | 31 | 32 | VDD_ADC |
| AIN4 | 33 | 34 | GND_ADC |
| AIN6 | 35 | 36 | AIN5 |
| AIN2 | 37 | 38 | AIN3 |
| AIN0 | 39 | 40 | AIN1 |
| GPIO20 | 41 | 42 | ECAPWMO |
| DGND | 43 | 44 | DGND |
| DGND | 45 | 46 | DGND |


| leyenda |
|---|
| Power, Ground, Reset |
| Pines digitales |
| Salida PWM |
| Entradas analógicas 1,8V |
| Bus I2C |
| Pines digitales reconfigurables |

**P8**

| Función | Pines físicos | | Función |
|---|---|---|---|
| DGND | 1 | 2 | DGND |
| MMC1_DAT6 | 3 | 4 | MMC1_DAT7 |
| MMC1_DAT2 | 5 | 6 | MMC1_DAT3 |
| GPIO_66 | 7 | 8 | GPIO_67 |
| GPIO_69 | 9 | 10 | GPIO_68 |
| GPIO_45 | 11 | 12 | GPIO_44 |
| EHRPWM2B | 13 | 14 | GPIO_26 |
| GPIO_47 | 15 | 16 | GPIO_46 |
| GPIO_27 | 17 | 18 | GPIO_65 |
| EHRPWM2A | 19 | 20 | MMC1_CMD |
| MMC1_CLK | 21 | 22 | MMC1_DAT5 |
| MMC1_DAT4 | 23 | 24 | MMC1_DAT1 |
| MMC1_DAT0 | 25 | 26 | GPIO_61 |
| LCD_VSYNC | 27 | 28 | LCD_PCLK |
| LCD_HSYNC | 29 | 30 | LCD_AC_BIAS |
| LCD_DATA14 | 31 | 32 | LCD_DATA15 |
| LCD_DATA13 | 33 | 34 | LCD_DATA11 |
| LCD_DATA12 | 35 | 36 | LCD_DATA10 |
| LCD_DATA8 | 37 | 38 | LCD_DATA9 |
| LCD_DATA6 | 39 | 40 | LCD_DATA7 |
| LCD_DATA4 | 41 | 42 | LCD_DATA5 |
| LCD_DATA2 | 43 | 44 | LCD_DATA3 |
| LCD_DATA0 | 45 | 46 | LCD_DATA1 |

Figura F.1: Diagrama de pines de la placa BBB

# G

# Aplicación de las reglas de Ziegler-Nichols

## G.1. Selección de parámetros para el controlador PID

Para la selección de los parámetros del controlador PID es necesario recurrir a las reglas de Ziegle-Nichols que fueron abordadas en la sección 2.3.4. En este caso, esto se debería hacer ya que como se ha visto en 4.8 el comportamiento del sensor no tiene un modelo matemático que sea capaz de definir la planta y por tanto esto conlleva a que encontrar los valores del controlador PID no sea posible a través de métodos analíticos.

El primer paso sera aplicar las reglas descritas en la sección 2.3.4 y comprobar si el comportamiento del sensor cumple los requerimientos de alguna de las reglas para así poder seleccionar los parámetros del controlador PID descrito en las tablas 2.3 o 2.4.

### G.1.1. Aplicación del método basado en la curva de reacción

En este primer método lo que se hace es tratar la planta como un sistema en lazo abierto en el cual se aplicara una entrada que sea el escalón unitario y la salida se debería obtener una respuesta en forma de S según se explica en la sección 2.3.4.1. En el caso de nuestro sensor la entrada sera la temperatura que se aplica para calentar el sensor y la salida la respuesta del sensor ante la presencia de odorante.

Por tanto, para simular la entrada ante el escalón unitario lo que se hará es hacer un cambio en la temperatura de calentamiento del sensor de 0 % al 100 % manteniendo este valor durante un periodo de tiempo para ver cual es el comportamiento de la salida. En este caso esta prueba se hará tanto para el aire, como para los odorantes de la muestra #0 definidos en la tabla 4.3.

#### G.1.1.1. Pasos a seguir para el experimento del método basado en la curva de reacción

Los pasos para realizar este experimento son los siguientes:

1. Antes de encender la BBB comprobar que esta la tarjeta microSD en la ranura correspondiente, ya que en esta unidad es donde se almacenarán los datos de los experimentos que

se realicen.

2. Comprobar que el sensor TGS2600 esta conectado en la placa de experimentación en el conector correspondiente para la modulación en amplitud, en este caso sobre el conector que tiene la serigrafía "TGS_AMP" sobre la placa.

3. Revisar las conexiones que hay entre las placas del circuito de protección, placa de experimentación y la BBB. En la tabla 3.5 se encuentran los pines que se deben comprobar que esta conectados correctamente.

4. Revisar que el cable de Ethernet este conectado a la BBB.

5. Conectar la fuente de alimentación de 5 V a la BBB, y esperar hasta que esta este encendida y se puede acceder. Una vez dentro situarse sobre el directorio /home/debian que es donde se encuentra el código para lanzar el experimento.

6. A continuación conectar la fuente externa de 24 V a la placa de experimentación.

7. Una vez estén las dos placas encendidas, lo primero que se hará es apagar el pin de la BBB que controla la succión del motor ya que este pin esta encendido por defecto y por tanto el motor estará funcionado. Para apagar el pin correspondiente sera necesario ejecutar el siguiente comando:

   python3 inicio.py

8. Ahora se ejecutara el siguiente comando para llevar a cabo el experimento para la modulación en regresión de temperatura:

   python3 impulso.py succion tiempo valvula

   En donde succión, tiempo y válvula son parámetros que debe pasar el usuario para llevar a cabo el experimento. El parámetro de succión es un valor entre el 65 % y 70 % y representa la succión del motor. El tiempo es un entero que representa el tiempo en minutos en el cual se va aplicar la máxima temperatura de calentamiento al sensor. Válvula es un entero que representa la electroválvula que va a estar activa para el experimento. El código del experimento se puede encontrar en el anexo I.4

9. Una vez terminado el experimento se recomienda apagar la plataforma de experimentación si esta no va hacer usada. Para apagar la BBB se lanzara el siguiente comando:

   shutdown -h now

   Se debe esperar hasta que la placa se apague, esto se comprobara por que los LEDs de la BBB se apagarán.

10. Una vez paga la BBB es conveniente desconectar primero la fuente de alimentación de 24 V de la placa de experimentación y a continuación la fuente de alimentación de 5 V de la BBB.

#### G.1.1.2. Experimento del método basado en la curva de reacción

A continuación se siguen los paso explicados en G.1.1.1 hasta el punto 8 en el cual lanzaremos el siguiente comando por consola:

```
python3 impulso.py 70 10 4
```

En este caso nuestro experimento tendrá una succión del motor del 70 % del tiempo del ciclo PWM, la temperatura máxima de calentamiento del sensor estará durante un tiempo de 10 minutos y la electroválvula que esta activa sera la número 4, en este caso esta se corresponde al aire para la primera versión de la placa. En la figura G.2, se observa la respuesta del aire ante una entrada que se ha simulado como si fuese el escalón unitario con un cambio temperatura desde su valor mínimo al máximo. Como se puede observar la respuesta del sensor ante una entrada tipo impulso genera una salida como la que se puede muestra en la figura 2.9.

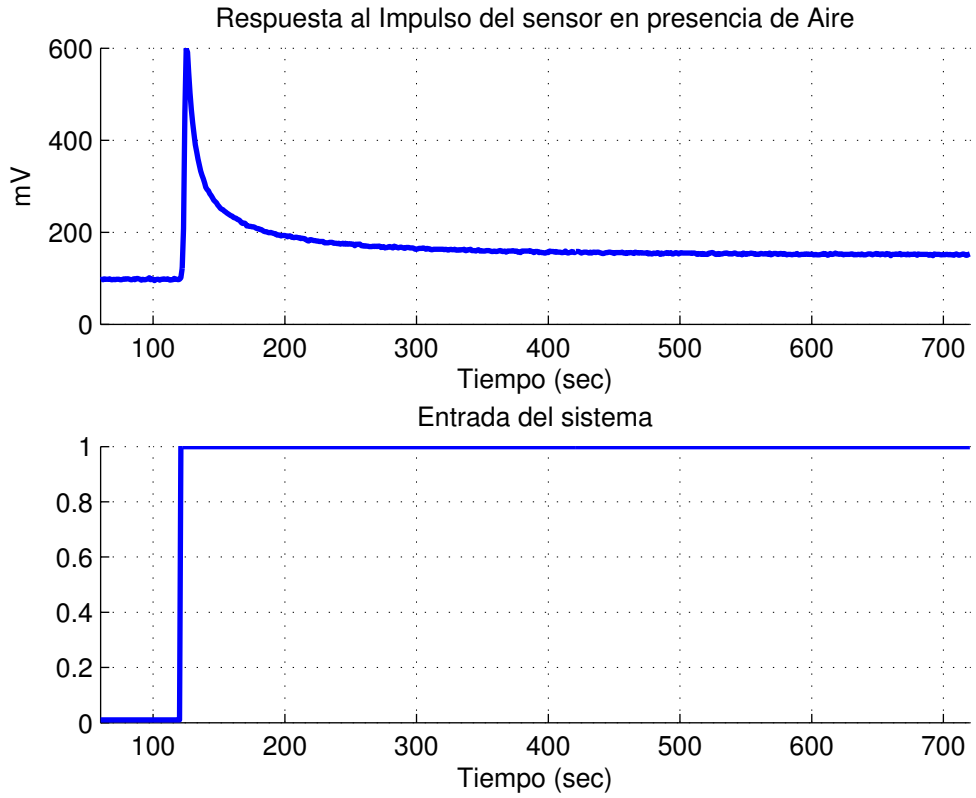



Figura G.1: Respuesta del sensor ante una entrada simulada como el escalón unitario ante la presencia de aire.

El experimento anterior también se ha realizado para los otros odorantes, teniendo como entrada del sistema la misma señal que se ha usado en la figura G.1. En la figura G.2, se puede observar la respuesta del sensor ante el impulso para el etanol, butano y metanol.

Como se puede observar de las figuras G.1 y G.2, la salida del sistema a simple vista parece no ser la esperada como se explica en 2.3.4.1 y por tanto no se podrían usar los parámetros de ajuste que de la tabla 2.3. Haciendo una ampliación de las figuras representadas anteriormente en la figura G.3, se puede observar como la salida tiene la forma de S descrita en la sección 2.3.4.1, con lo cual se podría usar este método para calcular los parámetros del controlador PID. Por otra parte, se puede observar que la salida es completamente diferente para cada odorante, y por tanto la curva en S que se espera va a variar de un odorante a otro. Esto implica que usar este método para calcular los parámetros del controlador puede resultar tedioso.

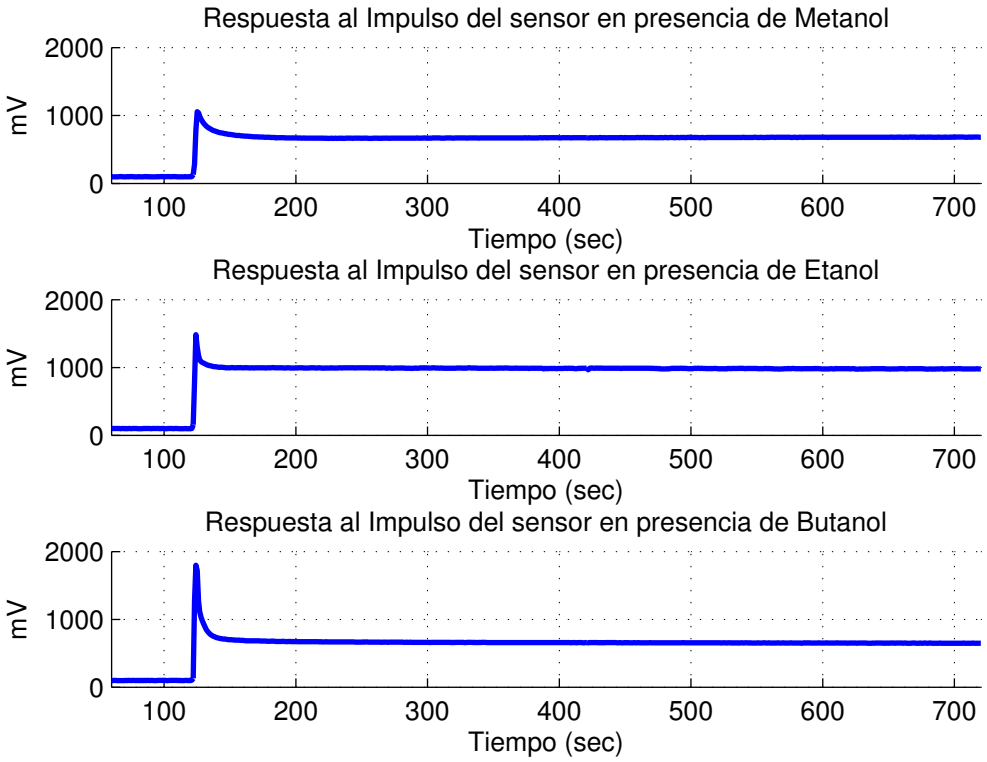



Figura G.2: Respuesta del sensor ante una entrada simulada como el escalón unitario ante la presencia de los odorantes bajo estudio.

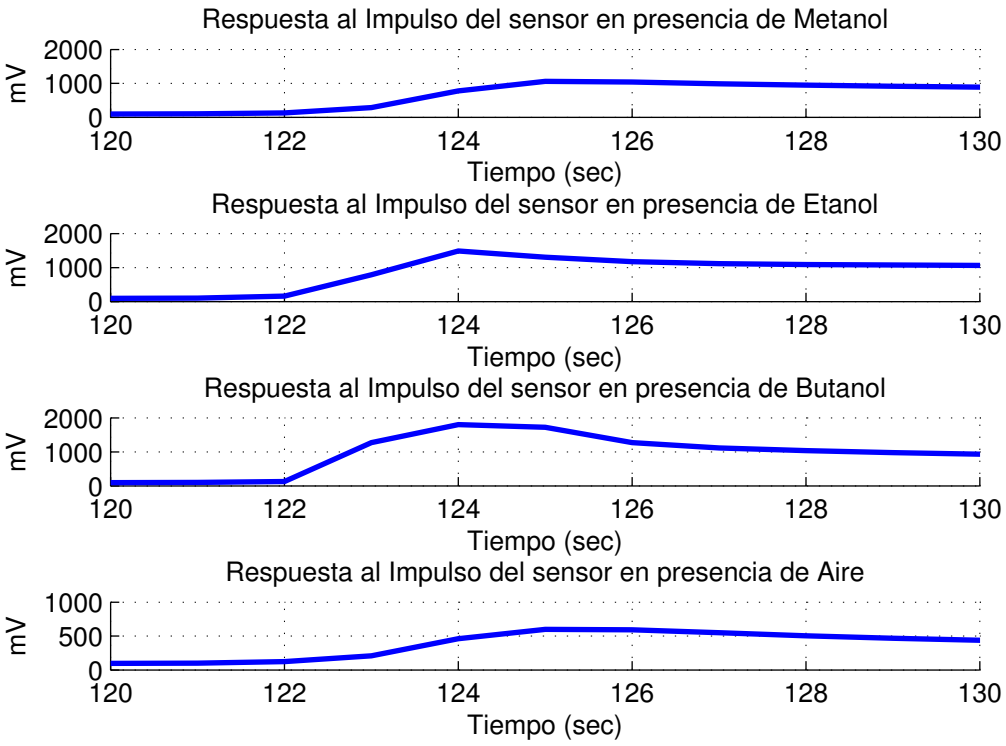



Figura G.3: Respuesta del sensor ante una entrada simulada como el escalón unitario ante la presencia de los odorantes bajo estudio.

# H

# Preparación de muestras

En el desarrollo de este trabajo ha sido importante el uso de muestras para llevar a cabo cada uno de los experimentos. Por tanto, a continuación se explicara como se deben preparar estas muestras y los elementos necesarios para proceder.

Antes de empezar con la preparación de la muestras sera necesario tener todos los materiales para la preparación de las muestras. Los materiales usados son los siguientes:

- Odorantes para la preparación de las muestras, en este caso metanol. etanol y butanol. En la figura H.1, se observan los odorantes usados.
- Tubos de ensayo, estos tubos son usados para almacenar cada una de las muestras. Se recomienda etiquetar cada uno de los tubos con el nombre del odorante y su respectiva concentración como se muestra en la figura H.2(a)
- Probetas, figura H.2(b). Se recomienda usar dos probetas, una de ellas la cual contendrá el agua destilada y la otra el odorante con el que se va a trabajar.
- Pipetas, estos elementos son usados para poder medir volúmenes de líquidos con gran precisión. Para la elaboración de las muestras serán necesario usar dos pipetas, una de ellas la cual es usada para medir volúmenes en mililitros figura H.2(c), y la otra para medir volúmenes en microlitros. figura H.2(d).

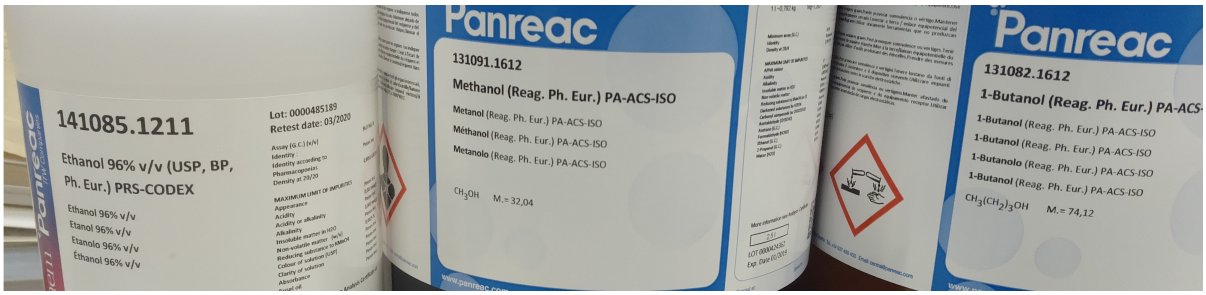
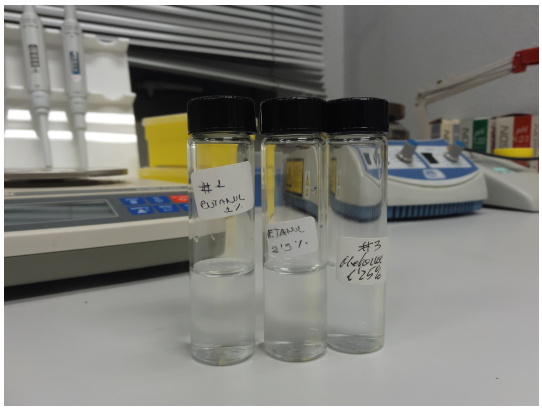
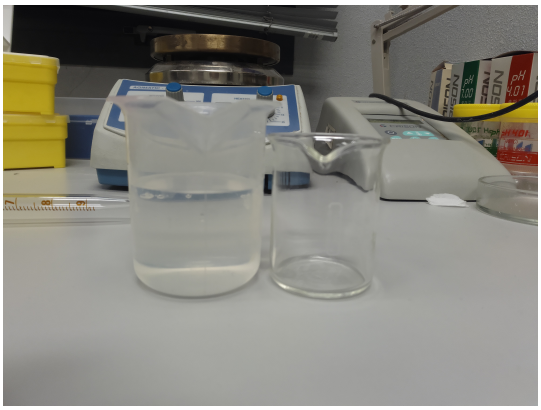



Figura H.1: Odorantes usados en el proyecto. Etanol, concentración del 96 %. Butanol. concentración del 99,8 %. Metanol, concentración del 99,8 %.

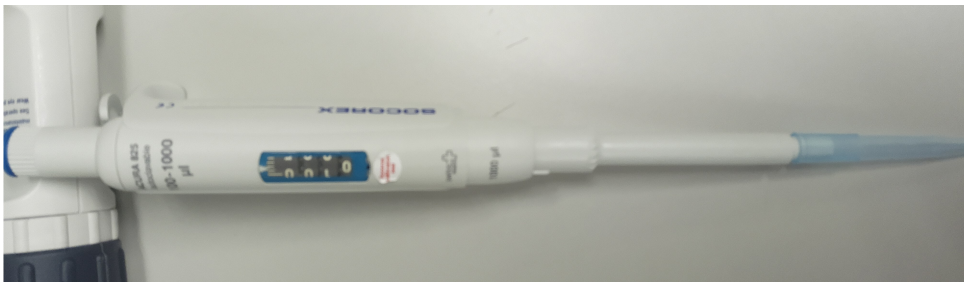


(a)



(b)

(c)



(d)

Figura H.2: Elementos de laboratorio para la preparación de muestras: (a)Tubos de ensayo. (b) Probetas. (c) Pipeta con precisión en mililitros. (d) Pipeta con precisión en microlitros.

A continuación, se explicará la forma en la que se debe preparar las muestras de odorantes con un ejemplo de una muestra de butanol con una concentración de 1 %. Los pasos que se deben seguir son los siguientes:

1. Primero se debe tener en cuentas las cantidades de agua y odorante que se deben usar para la mezcla, teniendo en cuenta que la mezcla será sobre un volumen total de 10 ml de disolución. En este caso, al ser una mezcla de 1 % de butanol, 9,9 ml se corresponden a agua y 0,1 ml (100 ul) a butanol.

2. En una de las probetas se vierte la suficiente cantidad de agua destilada para la mezcla, y en la otra probeta se vierte un poco del odorante. Si se van a realizar muestras de todos los odorantes se recomienda empezar primero por el metanol ya que este tiene un olor menos fuerte y al lavar la probeta correspondiente no dejara un fuerte olor impregnado sobre esta. Luego se puede seguir con el etanol y por último el butanol que tiene el olor más fuerte.

3. Usando la pipeta con precisión en mililitros se cogen los primeros 9 ml de agua destilada y se vierte sobre el tubo de ensayo. Luego usando la pipeta de precisión en microlitros se cogen los otros 900 ul para así completar la proporción de agua destilada que contiene la muestra.

4. Usando de nuevo la pipeta de precisión en microlitros se debe coger la cantidad correspondiente al odorante para completar así la muestra y depositarla en el tubo de ensayo.

5. Por último, una vez este la muestra en el tubo de ensayo esta se debe agitar para que se mezclen las disolución del agua y el odorante.

Para usar la pipeta de precisión en microlitros, lo primero que se debe hacer es ajustarla cantidad de líquido que se quiere obtener. Para ello la pipeta está dotada de un sistema de calibración manual en la que se debe poner la cantidad de microlitros. Una vez calibrada se debe primero presionar sobre la parte superior del cabezal hasta encontrar un punto en el que hay un bloqueo, en este momento la punta de la pipeta se debe introducir en el líquido manteniendo presionado el cabezal. Cuando la punta de la pipeta está en el líquido se debe soltar el cabezal y se podrá observar cómo se absorbe liquido en la punta. Por último, para soltar el líquido obtenido basta con presionar el cabezal de la pipeta hasta el final.

Para usar la pipeta de precisión en mililitros, esta se debe introducir en el liquido a obtener. Una vez de dentro se presiona sobre el cabezal hasta obtener la cantidad de liquido deseada. En este caso la pipeta tiene una resolución de 10 ml, y la medida empieza desde 10 ml desde la parte inferior de la pipeta hasta los 0 ml en la parte superior. Por tanto, si que quieren obtener 9 ml, se debe mantener presionado la pipeta hasta llegar a la parte superior a la linea marcada con 1 ml, de esta forma se obtiene 9 ml de liquido.

# I

# Códigos de ejecución

## I.1. Código sensor DHT22

```
1
import Adafruit_BBIO.ADC as ADC
3 import Adafruit_BBIO.GPIO as GPIO
import Adafruit_BBIO.PWM as PWM
5 import Adafruit_DHT
import sys
7 import os
import time
9 import math
import random
11 import numpy
from datetime import datetime, date
13 import _thread
##import PID
15
from  math import sin
17 from  math import pi
##import matplotlib.pyplot as plt
19
## Asignacion de puerto
21 sensorTemp22 = Adafruit_DHT.DHT22

23 Temp22       = 'P8_11' #Pin de lectura de temperatura/Humedad con DHT22 (Puerto
    GPIO_45)


25
motorPin     = 'P9_21' #Motor de succion
27 heatPin2600   = 'P9_22' #Calentamiento sensor TGS2600 (Puerto GPIO_2)
sensorPin2600 = 'P9_40' #Sensor TGS2600 (Puerto AIN_1)
29
## DECLARACION DE SALIDAS DEL SISTEMA.
31


33 ## DECLARACION VARIABLES DEL SISTEMA

35 SLEEP = 1      ## Periodo de estabilizacion  para cada adquisicion
target = []      ## Almaceno los valores de la senal objetivo
```

```
periodo = 40       ## Periodo de la senal Target (segundos)
error = 0.05     ## Parametro para controlar el tracking en %
temperaturaMax = 80 ## Ciclo pwm maximo temperatura sensor
temperaturaMin = 10 ## Cilco pwm minimo temperatura sensor

muestrasTGS =[]    ## Almacena las muestras de odorantes
temperaturaTGS =[]  ## Almacena la temperatura de calentamiento  del sensor
rS_TGS = []       ## Almacena la resistencia del sensor
maximo = 0        ## Almaceno el valor maximo de la senal objetivo
minimo =0      ## Alamceno el valor minimo de la senal objetivo


## VARIABLES SENSOR
Rl = 470  #omh
Vc = 5    #Voltios

SLEEP_tyh = 59  ##Medida de temperatura y humedad

## VARIABLES PARA EL PID
Kp = 0.6/0.3          ## Constante proporcional
Kd = 50.0/8           ## Constante derivativa
Ki = 50.0/2           ## Constante integral
##pid = PID.PID(Kp, Ki, Kd) ##
##pid.setSampleTime(SLEEP)

## Datos para el fichero de salida
nombreArchivo = "pid_tgs2600.txt"
archivoDatos = "pid_tgs2600.dat"
archivoTyH = "TyH_tgs2600.data"
fileString = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime()) +'_'+
    nombreArchivo
fileStringDatos = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime()) +'_'+
    archivoDatos
fileStringTyH = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime()) +'_'+
    archivoTyH
actual = datetime.now()
ruta = "/media/microSD/temperatura/"+str(actual.month)+"/"+str(actual.day)+"/"
if not os.path.exists(ruta): os.makedirs(ruta)

print ('creo ruta')
rutaArchivo = ruta.strip()+str(fileString)
rutaArchivoDatos = ruta.strip()+str(fileStringDatos)
rutaArchivoTyH = ruta.strip()+str(fileStringTyH)

## Se crean los ficheros de datos y temperatura
##datos = open(rutaArchivoDatos,"w+")
tempH = open(rutaArchivoTyH,"w+")

def fileInfoTGS2600(vectorValvulas, succion, heat2600, tiempo):
  f = open(rutaArchivo,"w+")
  f.write ('
    /*****************************************************************/')
  f.write ('Plataforma de experimentacion BBB\n')
  f.write ('Carlos Velasco\n')
  f.write ('Sensor TGS2600\n')
  f.write ('Algoritmo con lazo cerrado\n')
  fecha = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime())
  f.write ('Fecha y hora de inicio: ' + str(fecha) + '\n')
  f.write ('Ruta del fichero: ' + str(ruta) + '\n')
  f.write ('Nombre archivo: ' + str(nombreArchivo) + '\n')
  f.write ('Nombre archivo de datos: ' + str(archivoDatos) + '\n')
  f.write ('Nombre archivo de TyH: ' + str(archivoTyH) + '\n')
  f.write ('Conmutacion entre electrovalvulas: ' +str(vectorValvulas)+ '\n\n')
```

```
    f.write ('Parametros requeridos para el experimento \n')
97  f.write ('Succion del motor (50-100) >>> '+ str(succion)+ '%\n')
    f.write ('Temperatura Promedio TGS2600 (50-100) >>> '+str(heat2600)+'% Pin PWM
      : ' +str(heatPin2600)+ '\n\n')
99  f.write ('Duracion del experimento: ' +str(tiempo)+ ' minutos\n')
    f.write ('
      /*****************************************************************/')
101 f.write ('DATOS OBTENIDOS:\n\n')

103 def sensorTyH(tiempo):
    j=0
105 tiempo = round(tiempo)
    time.sleep(9)
107 # proceso de lectura
    while(j<tiempo):
109   tickHT = time.time()
      humedad, temperatura = Adafruit_DHT.read_retry(sensorTemp22, Temp22)
111   ahora = datetime.now()
      tackHT = time.time()
113   ticktack = tackHT-tickHT

115   if ticktack >SLEEP_tyh:
        print ("Tiempo medicion H y T > SLEEP:", ticktack)
117   else:
        print ("Tiempo medicion H y T:",ticktack)
119     if humedad is not None and temperatura is not None:
          print ('\nSensor DHT22: ' +str(sensorTemp22))
121       print ('>>> '+str(ahora)+' Temp = ' +str(temperatura)+ ' Humidity = ' +
      str(humedad)+'\n')
          tempH = open(rutaArchivoTyH, "a")
123       wline_h = str(ahora)+'\t'+str(temperatura)+'\t'+str(humedad)+'\n'
          tempH.writelines(wline_h)
125       tempH.flush()
        else:
127       print ('Failed to get reading, Try again!')
          tempH = open(rutaArchivoTyH, "a")
129       wline_h=str(ahora)+' '+str(temperatura)+' '+str(humedad)+'\n'
          tempH.writelines(wline_h)
131       tempH.flush()

133   tack=time.time()
      p = tack - tickHT;
135   time.sleep(SLEEP_tyh-p)
      j+=1
137


139


141 def finExperimento():
    ##PWM.stop(heatPin2600)
143 ##PWM.stop(motorPin)
    ##PWM.cleanup()
145 _thread.exit()
    ##fecha_fin=datetime.now()
147 print('\nEXPERIMENTO FINALIZADO CON EXITO')
    ##print 'Experimento terminado: '+str(fecha_fin)
149 ##f = open(ruta_fichero, "a")
    ##f.write('\nEXPERIMENTO FINALIZADO CON EXITO\n')
151 ##f.write('Fecha y hora de fin de experimentacion: ' +str(fecha_fin))
    ##f.flush()
153 ##f.close()
    ##datos.close()
155 tempH.close()
```

```
157 def main():

159   print ('*********************************************************\n'\
      '          ALGORITMO DE MODULACION EN LAZO CERRADO CONTROL PID \n'\
161   '*********************************************************\n' )

163   if(len(sys.argv)<2):
        print ('\n NUMERO DE PARAMETROS INCORRECTOS\n' \
165     'Los parametros que se deben pasar son:\n'\
        'Succion del motor (65% -100%)\n'
167     'Temperatura inicial TGS2600 (50%- 100%)\n'\
        'Tiempo de experimentacion (minutos)\n'\
169     'Tiempo de conmutacion  electrovalvulas (segundos)\n')
        return 0
171

173   ## Duracion del experimento
      tiempo = float(sys.argv[1])
175   if tiempo <1:
        print ('EL TIEMPO DE EXPERIMENTACION ES INCORECTO.\n' \
177     'Los parametros que se deben pasar son:\n'\
        'Succion del motor (50% -100%)\n'
179     'Temperatura inicial TGS2600 (50%- 100%)\n'\
        'Tiempo de experimentacion (minutos)\n'\
181     'Tiempo de conmutacion  electrovalvulas (segundos)\n')

183

185   ## Vector que contiene el orden de conmutacion de las electrovalvulas
      ##vec_open_valve = numpy.random.randint(1,5,tiempo*60/cambio)
187
      ## Llamada al hilo para la medida de temperatura y humedad
189   _thread.start_new_thread(sensorTyH,(tiempo,))

191   ## Se configura los puertos ADC, PWM

193

195   ## Sistema de control
      contador = 1
197   muestra = 0;

199

201   while contador < tiempo*60:
          tick = time.time()
203       tack = time.time();
          time.sleep(SLEEP-(tack-tick))
205       contador+=1

207   finExperimento()
      return 0
209
    if __name__ == '__main__':
211   try:
        main()
213   except KeyboardInterrupt:
        print ('Interrupted')
215     cierre()
        sys.exit(0)
```

resources/temperatura.py

## I.2. Código para la modulación pura usado en 4.7.1

```
import Adafruit_BBIO.ADC as ADC
import Adafruit_BBIO.GPIO as GPIO
import Adafruit_BBIO.PWM as PWM
import Adafruit_DHT
import sys
import os
import time
import math
import random
import numpy
from datetime import datetime, date
import _thread
##import PID

from  math import sin
from  math import pi
##import matplotlib.pyplot as plt

## Asignacion de puerto
sensorTemp22 = Adafruit_DHT.DHT22

Temp22      = 'P8_11' #Pin de lectura de temperatura/Humedad con DHT22 (Puerto GPIO_45)
electrovalve1 = 'P8_10' #Electrovalvula 1 (METANOL)
electrovalve2 = 'P8_12' #Electrovalvula 2 (ETANOL)
electrovalve3 = 'P8_14' #Electrovalvula 3 (BUTANOL)
electrovalve4 = 'P8_16' #Electrovalvula 4 (AIRE)

motorPin    = 'P9_21' #Motor de succion
heatPin2600   = 'P9_22' #Calentamiento sensor TGS2600 (Puerto GPIO_2)
sensorPin2600 = 'P9_40' #Sensor TGS2600 (Puerto AIN_1)

## DECLARACION DE SALIDAS DEL SISTEMA.
GPIO.setup("P8_10",GPIO.OUT)
GPIO.setup("P8_12",GPIO.OUT)
GPIO.setup("P8_14",GPIO.OUT)
GPIO.setup("P8_16",GPIO.OUT)

## DECLARACION VARIABLES DEL SISTEMA

SLEEP = 1     ## Periodo de estabilizacion  para cada adquisicion
target = []     ## Almaceno los valores de la senal objetivo
periodo = 40      ## Periodo de la senal Target (segundos)
error = 0.05    ## Parametro para controlar el tracking en %
temperaturaMax = 80 ## Ciclo pwm maximo temperatura sensor
temperaturaMin = 10 ## Cilco pwm minimo temperatura sensor

muestrasTGS =[]   ## Almacena las muestras de odorantes
temperaturaTGS =[]  ## Almacena la temperatura de calentamiento  del sensor
rS_TGS = []     ## Almacena la resistencia del sensor
maximo = 0      ## Almaceno el valor maximo de la senal objetivo
minimo =0     ## Alamceno el valor minimo de la senal objetivo


## VARIABLES SENSOR
Rl = 470  #omh
Vc = 5    #Voltios

SLEEP_tyh = 59  ##Medida de temperatura y humedad

## VARIABLES PARA EL PID
```

```
Kp = 0.6/0.3          ## Constante proporcional
Kd = 50.0/8           ## Constante derivativa
Ki = 50.0/2           ## Constante integral
##pid = PID.PID(Kp, Ki, Kd) ##
##pid.setSampleTime(SLEEP)

## Datos para el fichero de salida
nombreArchivo = "puro_tgs2600.txt"
archivoDatos =  "puro_tgs2600.dat"
archivoTyH = "puroTyH_tgs2600.data"
fileString = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime()) +'_'+
    nombreArchivo
fileStringDatos = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime()) +'_'+
    archivoDatos
fileStringTyH = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime()) +'_'+
    archivoTyH
actual = datetime.now()
ruta = "/media/microSD/PURO/"+str(actual.month)+"/"+str(actual.day)+"/"
if not os.path.exists(ruta): os.makedirs(ruta)


rutaArchivo = ruta.strip()+str(fileString)
rutaArchivoDatos = ruta.strip()+str(fileStringDatos)
rutaArchivoTyH = ruta.strip()+str(fileStringTyH)

## Se crean los ficheros de datos y temperatura
f = open(rutaArchivo,"w+")
datos = open(rutaArchivoDatos,"w+")
tempH = open(rutaArchivoTyH,"w+")

def fileInfoTGS2600(vectorValvulas, succion, heat2600, tiempo):
  f = open(rutaArchivo,"w+")
  f.write ('/*****************************************************************/
    ')
  f.write ('\nPlataforma de experimentacion BBB\n')
  f.write ('Carlos Velasco\n')
  f.write ('Sensor TGS2600\n')
  f.write ('Algoritmo Puro sin Modulacion\n')
  fecha = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime())
  f.write ('Fecha y hora de inicio: ' + str(fecha) + '\n')
  f.write ('Ruta del fichero: ' + str(ruta) + '\n')
  f.write ('Nombre archivo: ' + str(fileString) + '\n')
  f.write ('Nombre archivo de datos: ' + str(fileStringDatos) + '\n')
  f.write ('Nombre archivo de TyH: ' + str(fileStringTyH) + '\n')
  f.write ('Conmutacion entre electrovalvulas: ' +str(vectorValvulas)+ '\n\n')
  f.write ('Parametros requeridos para el experimento \n')
  f.write ('Succion del motor (65-100) >>> '+ str(succion)+ '%\n')
  f.write ('Temperatura Promedio TGS2600 (50-100) >>> '+str(heat2600)+'% Pin PWM
    : ' +str(heatPin2600)+ '\n\n')
  f.write ('Duracion del experimento: ' +str(tiempo)+ ' minutos\n')
  f.write ('/*****************************************************************/'
    )
  f.write ('\nDATOS OBTENIDOS:\n\n')

def sensorTyH(tiempo):
  j=0
  tiempo = round(tiempo)
  time.sleep(9)
  # proceso de lectura
  while(j<tiempo):
    tickHT = time.time()
    humedad, temperatura = Adafruit_DHT.read_retry(sensorTemp22, Temp22)
    ahora = datetime.now()
```

```
    tackHT = time.time()
119     ticktack = tackHT-tickHT

121     if ticktack >SLEEP_tyh:
          print ("Tiempo medicion H y T > SLEEP:", ticktack)
123     else:
          print ("Tiempo medicion H y T:",ticktack)
125       if humedad is not None and temperatura is not None:
            print ('\nSensor DHT22: ' +str(sensorTemp22))
127         print ('>>> '+str(ahora)+' Temp = ' +str(temperatura)+ ' Humidity = ' +
    str(humedad)+'\n')
            tempH = open(rutaArchivoTyH, "a")
129         wline_h = str(ahora)+'\t'+str(temperatura)+'\t'+str(humedad)+'\n'
            tempH.writelines(wline_h)
131         tempH.flush()
          else:
133         print ('Failed to get reading, Try again!')
            tempH = open(rutaArchivoTyH, "a")
135         wline_h=str(ahora)+' '+str(temperatura)+' '+str(humedad)+'\n'
            tempH.writelines(wline_h)
137         tempH.flush()

139     tack=time.time()
        p = tack - tickHT;
141     time.sleep(SLEEP_tyh-p)
        j+=1
143

145 """*****************************************************************
                        FUNCIONES DEL SISTEMA
147 *****************************************************************"""

149 def abrirElectrovalvula(electrovalvula):
      if electrovalvula ==1:
151     print ('electrovalvula 1\n')
        GPIO.output(electrovalve1, GPIO.LOW)
153     GPIO.output(electrovalve2, GPIO.HIGH)
        GPIO.output(electrovalve3, GPIO.HIGH)
155     GPIO.output(electrovalve4, GPIO.HIGH)
      elif electrovalvula == 2:
157     print ('electrovalvula 2\n')
        GPIO.output(electrovalve1, GPIO.HIGH)
159     GPIO.output(electrovalve2, GPIO.LOW)
        GPIO.output(electrovalve3, GPIO.HIGH)
161     GPIO.output(electrovalve4, GPIO.HIGH)
      elif electrovalvula == 3:
163     print ('electrovalvula 3\n')
        GPIO.output(electrovalve1, GPIO.HIGH)
165     GPIO.output(electrovalve2, GPIO.HIGH)
        GPIO.output(electrovalve3, GPIO.LOW)
167     GPIO.output(electrovalve4, GPIO.HIGH)
      elif electrovalvula == 4:
169     print ('electrovalvula 4\n')
        GPIO.output(electrovalve1, GPIO.HIGH)
171     GPIO.output(electrovalve2, GPIO.HIGH)
        GPIO.output(electrovalve3, GPIO.HIGH)
173     GPIO.output(electrovalve4, GPIO.LOW)

175 def cerrarElectrovalvula():
      GPIO.output(electrovalve1, GPIO.LOW)
177   GPIO.output(electrovalve2, GPIO.LOW)
      GPIO.output(electrovalve3, GPIO.LOW)
179   GPIO.output(electrovalve4, GPIO.LOW)
```

```
181 def motorStart(succion):
    PWM.start(motorPin,succion)
183
    def readADC():
185   ##valor =((1.8*(ADC.read_raw(sensorPin2600)))/4095)*1000
      valor = 0
187   valor = (1800*(ADC.read(sensorPin2600)))
      valor = round(valor,0)
189   return valor

191 def controlPuro(contador):

193   value = readADC()
      captura = datetime.now();
195   print ('Muestra puro['+str(contador)+']\tValor:'+str(value)+'\n')
      f = open(rutaArchivo,"a")
197   datos = open(rutaArchivoDatos,"a")
      wline_datos = str(contador)+'\t'+str(value)+'\t'+str(captura)+'\n'
199   wline_f = 'Muestra['+str(contador)+']\tValor:'+str(value)+'\t'+str(captura)+'\n
        '
      datos.writelines(wline_datos)
201   datos.flush()
      f.writelines(wline_f)
203   f.flush()


205
    def finExperimento():
207   PWM.stop(heatPin2600)
      PWM.stop(motorPin)
209   PWM.cleanup()
      _thread.exit()
211   cerrarElectrovalvula()
      fecha_fin=datetime.now()
213   print('\nEXPERIMENTO FINALIZADO CON EXITO')
      print ('Experimento terminado: '+str(fecha_fin))
215   f = open(rutaArchivo, "a")
      f.write('\nEXPERIMENTO FINALIZADO CON EXITO\n')
217   f.write('Fecha y hora de fin de experimentacion: ' +str(fecha_fin))
      f.flush()
219   f.close()
      datos.close()
221   tempH.close()

223 def main():

225   print ('************************************************************\n'\
      '          ALGORITMO DE MODULACION EN LAZO CERRADO CONTROL PID \n'\
227   '************************************************************\n')

229   if(len(sys.argv)<6):
        print ('\n NUMERO DE PARAMETROS INCORRECTOS\n' \
231     'Los parametros que se deben pasar son:\n'\
        'Succion del motor (65% -100%)\n'
233     'Temperatura inicial TGS2600 (50%- 100%)\n'\
        'Tiempo de experimentacion (minutos)\n'\
235     'Tiempo de conmutacion  electrovalvulas (segundos)\n'\
        'Electrovalvula incorrecta, valor entre 1 y 4\n')
237     return 0

239   ## Succion del motor 650-100%
      succion = float(sys.argv[1])
241   if succion>100 or succion<65:
```

```
    print ('PARAMETRO DE SUCCION DE MOTOR INCORRECTO.\n' \
243     'Los parametros que se deben pasar son:\n'\
    'Succion del motor (50% -100%)\n'
245     'Temperatura inicial TGS2600 (50%- 100%)\n'\
    'Tiempo de experimentacion (minutos)\n'\
247     'Tiempo de conmutacion  electrovalvulas (segundos)\n'\
    'Electrovalvula incorrecta, valor entre 1 y 4\n')
249     return 0

251   ## Temperatura media del sensor TGS2600
    heat2600 = float(sys.argv[2])
253   if heat2600 >100 or heat2600 <50:
    print ('PARAMETRO DE CALENTAMIENTO DEL SENSOR  INCORRECTO.\n' \
255     'Los parametros que se deben pasar son:\n'\
    'Succion del motor (50% -100%)\n'
257     'Temperatura inicial TGS2600 (50%- 100%)\n'\
    'Tiempo de experimentacion (minutos)\n'\
259     'Tiempo de conmutacion  electrovalvulas (segundos)\n'\
    'Electrovalvula incorrecta, valor entre 1 y 4\n')
261     return 0

263   ## Duracion del experimento
    tiempo = float(sys.argv[3])
265   if tiempo <1:
    print ('EL TIEMPO DE EXPERIMENTACION ES INCORECTO.\n' \
267     'Los parametros que se deben pasar son:\n'\
    'Succion del motor (50% -100%)\n'
269     'Temperatura inicial TGS2600 (50%- 100%)\n'\
    'Tiempo de experimentacion (minutos)\n'\
271     'Tiempo de conmutacion  electrovalvulas (segundos)\n'\
    'Electrovalvula incorrecta, valor entre 1 y 4\n')
273     return 0
    ## Tiempo conmutacion electrovalvula
275   cambio = float(sys.argv[4])
    if cambio <1:
277     print ('EL TIEMPO DE CONMUTACION DE VALVULAS ES INCORRECTO.\n' \
    'Los parametros que se deben pasar son:\n'\
279     'Succion del motor (50% -100%)\n'
    'Temperatura inicial TGS2600 (50%- 100%)\n'\
281     'Tiempo de experimentacion (minutos)\n'\
    'Tiempo de conmutacion  electrovalvulas (segundos)\n'\
283     'Electrovalvula incorrecta, valor entre 1 y 4\n')
    return 0
285   ## Valvula conmutar
    valvula = float(sys.argv[5])
287   if valvula<1 and valvula>5:
    print ('EL TIEMPO DE CONMUTACION DE VALVULAS ES INCORRECTO.\n' \
289     'Los parametros que se deben pasar son:\n'\
    'Succion del motor (50% -100%)\n'
291     'Temperatura inicial TGS2600 (50%- 100%)\n'\
    'Tiempo de experimentacion (minutos)\n'\
293     'Tiempo de conmutacion  electrovalvulas (segundos)\n'\
    'Electrovalvula incorrecta, valor entre 1 y 4\n')
295     return 0

297   valvula = int(valvula)
    ## Vector que contiene el orden de conmutacion de las electrovalvulas
299   ##vectorValvulas = numpy.random.randint(1,5,tiempo*60/cambio)
    vectorValvulas =[4,4,valvula,4,4,valvula,4,4,valvula,4,4,valvula,4,4,valvula
      ,4,4]
301   #vectorValvulas =[4,2, 4,2, 4,2,4,2, 4,2,4,2,4,2,4]
    ##vectorValvulas =[4,3, 4,3, 4,3,4,3, 4,3,4,3,4,3,4]
303   print ('Conmutacion electrovalvulas: '+ str(vectorValvulas)+'\n')
```

```
    ## Llamada al hilo para la medida de temperatura y humedad
305 _thread.start_new_thread(sensorTyH,(tiempo,))

307 ## Se configura los puertos ADC, PWM
    motorStart(succion)
309 ADC.setup()
    PWM.start(heatPin2600,heat2600)
311
    samples = 0
313 while samples < 60:
      tick = time.time()
315   print ('Medidas de estabilizacion\t'+str(readADC())) ## primera medida
      erronea
      samples+=1
317   tack = time.time();
      time.sleep(SLEEP-(tack-tick))
319
    ## creacion archivo de informacion experimento
321 fileInfoTGS2600(vectorValvulas, succion, heat2600, tiempo)
    ## Sistema de control
323 contador = 1
    muestra = 0;
325 pos = 0 ;

327 while contador <= tiempo*60:

329   electrovalvula = vectorValvulas[pos]
      if electrovalvula == 1:
331     print ('ELECTROVALVULA: '+str(electrovalvula)+' METANOL\n')
      elif electrovalvula == 2:
333     print ('ELECTROVALVULA: '+str(electrovalvula)+' ETANOL \n')
      elif electrovalvula == 3:
335     print ('ELECTROVALVULA: '+str(electrovalvula)+' BUTANOL \n')
      else:
337     print ('ELECTROVALVULA: '+str(electrovalvula)+' AIRE \n')

339   abrirElectrovalvula(electrovalvula)

341   j = 0;
      while j < cambio:
343     tick = time.time()
        controlPuro(contador)
345     contador+=1
        j+=1
347     tack = time.time();
        time.sleep(SLEEP-(tack-tick))
349   pos+=1

351 finExperimento()
    return 0
353
if __name__ == '__main__':
355 try:
      main()
357 except KeyboardInterrupt:
      print ('Interrupted')
359   finExperimento()
      sys.exit(0)
```

resources/puro4.py

## I.3. Código usado para llevar a cabo el estudio del sensor TG2600

```
import Adafruit_BBIO.ADC as ADC
import Adafruit_BBIO.GPIO as GPIO
import Adafruit_BBIO.PWM as PWM
import Adafruit_DHT
import sys
import os
import time
import math
import random
import numpy
from datetime import datetime, date
import _thread
import PID

from  math import sin
from  math import pi
##import matplotlib.pyplot as plt

## Asignacion de puerto
sensorTemp22 = Adafruit_DHT.DHT22

Temp22      = 'P8_11' #Pin de lectura de temperatura/Humedad con DHT22 (Puerto GPIO_45)
electrovalve1 = 'P8_10' #Electrovalvula 1 (METANOL)
electrovalve2 = 'P8_12' #Electrovalvula 2 (ETANOL)
electrovalve3 = 'P8_14' #Electrovalvula 3 (BUTANOL)
electrovalve4 = 'P8_16' #Electrovalvula 4 (AIRE)

motorPin    = 'P9_21' #Motor de succion
heatPin2600   = 'P9_22' #Calentamiento sensor TGS2600 (Puerto GPIO_2)
sensorPin2600 = 'P9_40' #Sensor TGS2600 (Puerto AIN_1)

## DECLARACION DE SALIDAS DEL SISTEMA.
GPIO.setup("P8_10",GPIO.OUT)
GPIO.setup("P8_12",GPIO.OUT)
GPIO.setup("P8_14",GPIO.OUT)
GPIO.setup("P8_16",GPIO.OUT)

## DECLARACION VARIABLES DEL SISTEMA

SLEEP = 1     ## Periodo de estabilizacion  para cada adquisicion
target = []     ## Almaceno los valores de la senal objetivo
periodo = 40      ## Periodo de la senal Target (segundos)
error = 0.05    ## Parametro para controlar el tracking en %
temperaturaMax = 80 ## Ciclo pwm maximo temperatura sensor
temperaturaMin = 10 ## Cilco pwm minimo temperatura sensor

muestrasTGS =[]   ## Almacena las muestras de odorantes
temperaturaTGS =[]  ## Almacena la temperatura de calentamiento  del sensor
rS_TGS = []     ## Almacena la resistencia del sensor
maximo = 0      ## Almaceno el valor maximo de la senal objetivo
minimo =0     ## Alamceno el valor minimo de la senal objetivo
tempActual =0
lectura = 10;


## VARIABLES SENSOR
Rl = 470  #omh
Vc = 5    #Voltios

SLEEP_tyh = 59  ##Medida de temperatura y humedad
```

```
62 ## VARIABLES PARA EL PID
   Kp = 0.6*0.3          ## Constante proporcional
64 Kd = 50.0/8           ## Constante derivativa
   Ki = 50.0/2           ## Constante integral
66



68
   ## Datos para el fichero de salida
70 nombreArchivo = "rampaAire_tgs2600.txt"
   archivoDatos = "rampaAire_tgs2600.dat"
72 ##archivoTyH = "pid_tgs2600TyH.data"
   fileString = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime()) +'_'+
       nombreArchivo
74 fileStringDatos = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime()) +'_'+
       archivoDatos
   ##fileStringTyH = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime()) +'_'+
       archivoTyH
76 actual = datetime.now()
   ruta = "/media/microSD/rampa/"+str(actual.month)+"/"+str(actual.day)+"/"
78 if not os.path.exists(ruta): os.makedirs(ruta)


80
   rutaArchivo = ruta.strip()+str(fileString)
82 rutaArchivoDatos = ruta.strip()+str(fileStringDatos)
   ##rutaArchivoTyH = ruta.strip()+str(fileStringTyH)
84
   ## Se crean los ficheros de datos y temperatura
86 f = open(rutaArchivo,"w+")
   datos = open(rutaArchivoDatos,"w+")
88 ##tempH = open(rutaArchivoTyH,"w+")

90 def fileInfoTGS2600(vectorValvulas, succion, heat2600, tiempo, target, odorante):
     f = open(rutaArchivo,"w+")
92   f.write ('
       /*****************************************************************/')
     f.write ('Plataforma de experimentacion BBB\n')
94   f.write ('Carlos Velasco\n')
     f.write ('Sensor TGS2600\n')
96   f.write ('PRECALENTAMIENTO SENSOR\n')
     fecha = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime())
98   f.write ('Fecha y hora de inicio: ' + str(fecha) + '\n')
     f.write ('Ruta del fichero: ' + str(ruta) + '\n')
100  f.write ('Nombre archivo: ' + str(fileString) + '\n')
     f.write ('Nombre archivo de datos: ' + str(fileStringDatos) + '\n')
102  ##f.write ('Nombre archivo de TyH: ' + str(fileStringTyH) + '\n')
     f.write ('Conmutacion entre electrovalvulas: ' +str(vectorValvulas)+ '\n\n')
104  f.write ('Parametros requeridos para el experimento \n')
     f.write ('Succion del motor (50-100) >>> '+ str(succion)+ '%\n')
106  f.write ('Temperatura Promedio TGS2600 (50-100) >>> '+str(heat2600)+'% Pin PWM
       : ' +str(heatPin2600)+ '\n\n')
     f.write ('Duracion del experimento: ' +str(tiempo)+ ' minutos\n')
108  f.write ('Odorante bajo prueba: ' +str(odorante)+ '\n')
     f.write ('
       /*****************************************************************/\n')
110  f.write ('Target Inicial:\n')
     f.write (str(target)+'\n')
112  f.write ('DATOS OBTENIDOS:\n\n')

114 """*****************************************************************
                          FUNCIONES DEL SISTEMA
116 *****************************************************************"""
```

```
118 def abrirElectrovalvula(electrovalvula):
    if electrovalvula ==1:
120     print ('electrovalvula 1\n')
        GPIO.output(electrovalve1, GPIO.LOW)
122     GPIO.output(electrovalve2, GPIO.HIGH)
        GPIO.output(electrovalve3, GPIO.HIGH)
124     GPIO.output(electrovalve4, GPIO.HIGH)
    elif electrovalvula == 2:
126     print ('electrovalvula 2\n')
        GPIO.output(electrovalve1, GPIO.HIGH)
128     GPIO.output(electrovalve2, GPIO.LOW)
        GPIO.output(electrovalve3, GPIO.HIGH)
130     GPIO.output(electrovalve4, GPIO.HIGH)
    elif electrovalvula == 3:
132     print ('electrovalvula 3\n')
        GPIO.output(electrovalve1, GPIO.HIGH)
134     GPIO.output(electrovalve2, GPIO.HIGH)
        GPIO.output(electrovalve3, GPIO.LOW)
136     GPIO.output(electrovalve4, GPIO.HIGH)
    elif electrovalvula == 4:
138     print ('electrovalvula 4\n')
        GPIO.output(electrovalve1, GPIO.HIGH)
140     GPIO.output(electrovalve2, GPIO.HIGH)
        GPIO.output(electrovalve3, GPIO.HIGH)
142     GPIO.output(electrovalve4, GPIO.LOW)

144 def motorStart(succion):
    PWM.start(motorPin,succion)
146
    def readADC():
148   valor = (1800*(ADC.read(sensorPin2600)))
      valor = round(valor,0)
150   return valor

152 def esperar(tiempo):
    time.sleep(tiempo)
154
    def mediaCalententamiento():
156   print ('Entro mediaCalententamiento')
      value = 0;
158   NM = 0;
      while NM <lectura:
160     tick = time.time()
        adcValue = readADC()
162     print ('adcValue: \t'+str(adcValue))
        value += adcValue
164     NM+=1
        tack = time.time();
166     time.sleep(SLEEP-(tack-tick))
      value/=lectura
168   value = round(value)
      print ('Media:\t'+str(value))
170   return value


172


174 """Funcion que se encarga de llevar a acabo el experimento"""
    def controlSistema(temperatura):
176   ##ADC.read_raw(sensorPin2600)  ## primera medida erronea
      value = readADC()
178   print ('value:\t'+str(value))
      captura = datetime.now();
180
```

```
    ## Caclulo del valor de resistencia interna del sensor
182 ## el valor devuelto por read_raw ata en mV por eso divide entre 1000
    ##RStgs2600 = ((Vc*Rl)/(value/1000.0)) - Rl
184

186 ## Se muestra por pantalla los datos
    print ('Temperatura['+str(temperatura)+']\tValor:'+str(value)+'mV\t'+str(
      captura)+'\n')
188
    ## Se guardan datos en archivo
190 f = open(rutaArchivo,"a")
    datos = open(rutaArchivoDatos,"a")
192 wline_datos = str(temperatura)+'\t'+str(value)+'\t'+str(captura)+'\n'
    wline_f = 'Temperatura['+str(temperatura)+'V >> Valor:'+str(value)+'mV '+str(
      captura)+'\n'
194 datos.writelines(wline_datos)
    datos.flush()
196 f.writelines(wline_f)
    f.flush()
198
  def finExperimento():
200 PWM.stop(heatPin2600)
    PWM.stop(motorPin)
202 PWM.cleanup()
    ##_thread.exit()
204
    fecha_fin=datetime.now()
206 print('\nEXPERIMENTO FINALIZADO CON EXITO')
    print ('Experimento terminado: '+str(fecha_fin))
208 f = open(rutaArchivo, "a")
    f.write('\nEXPERIMENTO FINALIZADO CON EXITO\n')
210 f.write('Fecha y hora de fin de experimentacion: ' +str(fecha_fin))
    f.flush()
212 f.close()
    datos.close()
214 ##tempH.close()

216 def main():

218 print ('*********************************************\n'\
    '        RAMPA DE CALENTAMIENTO \n'\
220 '*************************************************\n' )

222 if(len(sys.argv)<4):
      print ('\n NUMERO DE PARAMETROS INCORRECTOS\n' \
224   'Los parametros que se deben pasar son:\n'\
      'Succion del motor (65% -100%)\n'\
226   'Numero de ciclos de calentamiento (>1)\n')
      return 0
228
    ## Succion del motor 50-100%
230 succion = float(sys.argv[1])
    if succion>100 or succion<65:
232   print ('PARAMETRO DE SUCCION DE MOTOR INCORRECTO.\n' \
      'Los parametros que se deben pasar son:\n'\
234   'Succion del motor (65% -100%)\n')
      return 0
236
     ## Numero de ciclos calentamiento
238 numCiclos = float(sys.argv[2])
    if numCiclos<1:
240   print ('PARAMETRO DE SUCCION DE MOTOR INCORRECTO.\n' \
      'Los parametros que se deben pasar son:\n'\
```

```
242     'Succion del motor (65% -100%)\n')

244   ## Valvula que se quire abrirElectrovalvula
      odorante = int (float(sys.argv[3]))
246   if odorante < 1 or odorante >4:
        print ('VALVULA INCORRECTA.\n' \
248     'Los parametros que se deben pasar son:\n'\
        'Succion del motor (65% -100%)\n')
250     return 0


252
      ## Vector que contiene el orden de conmutacion de las electrovalvulas
254   ##vectorValvulas = numpy.random.randint(4,5,tiempo*60/cambio)
      ##vectorValvulas =[4,4,4]
256   print(succion)
      ## Se configura los puertos ADC, PWM
258   motorStart(succion)
      ADC.setup()
260   PWM.start(heatPin2600,95)


262
      samples = 0
264   while samples < 60:
        tick = time.time()
266     print ('Medidas de estabilizacion:\t'+str(readADC()))  ## primera medida
       erronea
        samples+=1
268     tack = time.time();
        time.sleep(SLEEP-(tack-tick))
270
      vectorT= range(0,100,5)#
        #[0,5,10,15,20,25,30,35,40,45,50,55,60,65,70,75,80,85,90,95]
272   vectorTd= range(95,-5,-5)
      ## creacion archivo de informacion experimento
274   fileInfoTGS2600(vectorT, succion, "95", "0", "0", odorante)

276   ## Sistema de control

278   contador = 0
      ciclo = 0
280   abrirElectrovalvula(odorante)
      while ciclo <numCiclos:
282     PWM.set_duty_cycle(heatPin2600,vectorTd[0])
        while contador<20:
284       tick = time.time()
          print("Temperatura:\t"+str(vectorTd[0]))
286       controlSistema(vectorTd[0])
          contador+=1
288       tack = time.time();
          time.sleep(SLEEP-(tack-tick))
290
        contador=0
292     while contador < len(vectorTd):
          PWM.set_duty_cycle(heatPin2600,vectorTd[contador])
294       tick = time.time()
          print("Temperatura:\t"+str(vectorTd[contador]))
296       controlSistema(vectorTd[contador])
          contador+=1
298       tack = time.time();
          time.sleep(SLEEP-(tack-tick))
300
        contador=0
302     PWM.set_duty_cycle(heatPin2600,vectorTd[19])
```

```
    while contador<20:
304       tick = time.time()
          print("Temperatura:\t"+str(vectorTd[19]))
306       controlSistema(vectorTd[19])
          contador+=1
308       tack = time.time();
          time.sleep(SLEEP-(tack-tick))
310
      contador=0
312   while contador < len(vectorT):
        PWM.set_duty_cycle(heatPin2600,vectorT[contador])
314       tick = time.time()
          print("Temperatura:\t"+str(vectorT[contador]))
316       controlSistema(vectorT[contador])
          contador+=1
318       tack = time.time();
          time.sleep(SLEEP-(tack-tick))
320
      contador=0
322   ciclo+=1

324 finExperimento()
    return 0
326
if __name__ == '__main__':
328 try:
      main()
330 except KeyboardInterrupt:
      print ('Interrupted')
332   finExperimento()
      sys.exit(0)
```

resources/rampaPre.py

## I.4. Código usado para aplicar el método basado en la curva de reacción

```
1 import Adafruit_BBIO.ADC as ADC
  import Adafruit_BBIO.GPIO as GPIO
3 import Adafruit_BBIO.PWM as PWM
  import Adafruit_DHT
5 import sys
  import os
7 import time
  import math
9 import random
  import numpy
11 from datetime import datetime, date
  import _thread
13 import PID

15 from  math import sin
  from  math import pi
17 ##import matplotlib.pyplot as plt

19 ## Asignacion de puerto
  sensorTemp22 = Adafruit_DHT.DHT22
21
  Temp22      = 'P8_11' #Pin de lectura de temperatura/Humedad con DHT22 (Puerto
      GPIO_45)
```

```
electrovalve1 = 'P8_10' #Electrovalvula 1 (METANOL)
electrovalve2 = 'P8_12' #Electrovalvula 2 (ETANOL)
electrovalve3 = 'P8_14' #Electrovalvula 3 (BUTANOL)
electrovalve4 = 'P8_16' #Electrovalvula 4 (AIRE)

motorPin     = 'P9_21' #Motor de succion
heatPin2600  = 'P9_22' #Calentamiento sensor TGS2600 (Puerto GPIO_2)
sensorPin2600 = 'P9_40' #Sensor TGS2600 (Puerto AIN_1)

## DECLARACION DE SALIDAS DEL SISTEMA.
GPIO.setup("P8_10",GPIO.OUT)
GPIO.setup("P8_12",GPIO.OUT)
GPIO.setup("P8_14",GPIO.OUT)
GPIO.setup("P8_16",GPIO.OUT)

## DECLARACION VARIABLES DEL SISTEMA

SLEEP = 1      ## Periodo de estabilizacion  para cada adquisicion
target = []      ## Almaceno los valores de la senal objetivo
periodo = 40       ## Periodo de la senal Target (segundos)
error = 0.05     ## Parametro para controlar el tracking en %
temperaturaMax = 80 ## Ciclo pwm maximo temperatura sensor
temperaturaMin = 10 ## Cilco pwm minimo temperatura sensor

muestrasTGS =[]   ## Almacena las muestras de odorantes
temperaturaTGS =[]  ## Almacena la temperatura de calentamiento  del sensor
rS_TGS = []      ## Almacena la resistencia del sensor
maximo = 0       ## Almaceno el valor maximo de la senal objetivo
minimo =0      ## Alamceno el valor minimo de la senal objetivo
tempActual =0
lectura = 10;

## VARIABLES SENSOR
Rl = 470  #omh
Vc = 5    #Voltios

SLEEP_tyh = 59  ##Medida de temperatura y humedad

## VARIABLES PARA EL PID
Kp = 0.6*0.3          ## Constante proporcional
Kd = 50.0/8           ## Constante derivativa
Ki = 50.0/2           ## Constante integral



## Datos para el fichero de salida
nombreArchivo = "impulso.txt"
archivoDatos = "impulso.dat"
##archivoTyH = "pid_tgs2600TyH.data"
fileString = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime()) +'_'+
    nombreArchivo
fileStringDatos = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime()) +'_'+
    archivoDatos
##fileStringTyH = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime()) +'_'+
    archivoTyH
actual = datetime.now()
ruta = "/media/microSD/impulso/"+str(actual.month)+"/"+str(actual.day)+"/"
if not os.path.exists(ruta): os.makedirs(ruta)


rutaArchivo = ruta.strip()+str(fileString)
rutaArchivoDatos = ruta.strip()+str(fileStringDatos)
##rutaArchivoTyH = ruta.strip()+str(fileStringTyH)
```

```
83
## Se crean los ficheros de datos y temperatura
85 f = open(rutaArchivo,"w+")
datos = open(rutaArchivoDatos,"w+")
87 ##tempH = open(rutaArchivoTyH,"w+")

89 def fileInfoTGS2600(valvula, succion, tiempo):
    f = open(rutaArchivo,"w+")
91  f.write ('/************************************************/')
    f.write ('Plataforma de experimentacion BBB\n')
93  f.write ('Carlos Velasco\n')
    f.write ('Sensor TGS2600\n')
95  f.write ('Respues al impulso del sensor\n')
    fecha = time.strftime("%a %d %b %Y - %H:%M:%S", time.localtime())
97  f.write ('Fecha y hora de inicio: ' + str(fecha) + '\n')
    f.write ('Ruta del fichero: ' + str(ruta) + '\n')
99  f.write ('Nombre archivo: ' + str(fileString) + '\n')
    f.write ('Nombre archivo de datos: ' + str(fileStringDatos) + '\n')
101  ##f.write ('Nombre archivo de TyH: ' + str(fileStringTyH) + '\n')
    f.write ('Electrovalvulas: ' +str(valvula)+ '\n\n')
103  f.write ('Parametros requeridos para el experimento \n')
    f.write ('Succion del motor (50-100) >>> '+ str(succion)+ '%\n')
105  f.write ('Duracion del experimento: ' +str(tiempo)+ ' minutos\n')
    f.write ('/*****************************************************/\n')
107  f.write ('Target Inicial:\n')
    f.write ('DATOS OBTENIDOS:\n\n')
109

"""*********************************************************************
111                        FUNCIONES DEL SISTEMA
*********************************************************************"""
113

def abrirElectrovalvula(electrovalvula):
115  if electrovalvula ==1:
      print ('electrovalvula 1\n')
117   GPIO.output(electrovalve1, GPIO.LOW)
      GPIO.output(electrovalve2, GPIO.HIGH)
119   GPIO.output(electrovalve3, GPIO.HIGH)
      GPIO.output(electrovalve4, GPIO.HIGH)
121  elif electrovalvula == 2:
      print ('electrovalvula 2\n')
123   GPIO.output(electrovalve1, GPIO.HIGH)
      GPIO.output(electrovalve2, GPIO.LOW)
125   GPIO.output(electrovalve3, GPIO.HIGH)
      GPIO.output(electrovalve4, GPIO.HIGH)
127  elif electrovalvula == 3:
      print ('electrovalvula 3\n')
129   GPIO.output(electrovalve1, GPIO.HIGH)
      GPIO.output(electrovalve2, GPIO.HIGH)
131   GPIO.output(electrovalve3, GPIO.LOW)
      GPIO.output(electrovalve4, GPIO.HIGH)
133  elif electrovalvula == 4:
      print ('electrovalvula 4\n')
135   GPIO.output(electrovalve1, GPIO.HIGH)
      GPIO.output(electrovalve2, GPIO.HIGH)
137   GPIO.output(electrovalve3, GPIO.HIGH)
      GPIO.output(electrovalve4, GPIO.LOW)
139

def motorStart(succion):
141  PWM.start(motorPin,succion)

143 def readADC():
    valor = (1800*(ADC.read(sensorPin2600)))
145  valor = round(valor,0)
```

```
  return valor

def esperar(tiempo):
  time.sleep(tiempo)

def mediaCalententamiento():
  print ('Entro mediaCalententamiento')
  value = 0;
  NM = 0;
  while NM <lectura:
    tick = time.time()
    adcValue = readADC()
    print ('adcValue: \t'+str(adcValue))
    value += adcValue
    NM+=1
    tack = time.time();
    time.sleep(SLEEP-(tack-tick))
  value/=lectura
  value = round(value)
  print ('Media:\t'+str(value))
  return value


"""Funcion que se encarga de llevar a acabo el experimento"""
def controlSistema(segundos, temperatura):
  ##ADC.read_raw(sensorPin2600)  ## primera medida erronea
  value = readADC()
  print ('value:\t'+str(value))
  captura = datetime.now();

  ## Caclulo del valor de resistencia interna del sensor
  ## el valor devuelto por read_raw ata en mV por eso divide entre 1000
  ##RStgs2600 = ((Vc*Rl)/(value/1000.0)) - Rl


  ## Se muestra por pantalla los datos
  print ('Muesta:'+str(segundos)+'\tTemperatura['+str(temperatura)+']\tValor:'+
    str(value)+'mV\t'+str(captura)+'\n')

  ## Se guardan datos en archivo
  f = open(rutaArchivo,"a")
  datos = open(rutaArchivoDatos,"a")
  wline_datos = str(segundos)+'\t'+str(temperatura)+'\t'+str(value)+'\t'+str(
    captura)+'\n'
  wline_f = 'Muesta:'+str(segundos)+'\tTemperatura['+str(temperatura)+'V >> Valor
    :'+str(value)+'mV '+str(captura)+'\n'
  datos.writelines(wline_datos)
  datos.flush()
  f.writelines(wline_f)
  f.flush()

def finExperimento():
  PWM.stop(heatPin2600)
  PWM.stop(motorPin)
  PWM.cleanup()
  ##_thread.exit()

  fecha_fin=datetime.now()
  print('\nEXPERIMENTO FINALIZADO CON EXITO')
  print ('Experimento terminado: '+str(fecha_fin))
  f = open(rutaArchivo, "a")
  f.write('\nEXPERIMENTO FINALIZADO CON EXITO\n')
  f.write('Fecha y hora de fin de experimentacion: ' +str(fecha_fin))
```

```
    f.flush()
207 f.close()
    datos.close()
209 ##tempH.close()

211 def main():

213 print ('*********************************************************\n'\
    '                Respuesta al impulso                    \n'\
215 '*********************************************************\n' )

217 if(len(sys.argv)<4):
      print ('\n NUMERO DE PARAMETROS INCORRECTOS\n' \
219   'Los parametros que se deben pasar son:\n'\
      'Succion del motor (65% -100%)\n'\
221   'Numero de ciclos de calentamiento (>1)\n')
      return 0
223
    ## Succion del motor 50-100%
225 succion = float(sys.argv[1])
    if succion>100 or succion<65:
227   print ('PARAMETRO DE SUCCION DE MOTOR INCORRECTO.\n' \
      'Los parametros que se deben pasar son:\n'\
229   'Succion del motor (65% -100%)\n')
      return 0
231
     ## timmpo de calentamiento
233 tiempo = float(sys.argv[2])
    if tiempo<0:
235   print ('PARAMETRO DE TIEMPO INCORRECTO.\n' \
      'Los parametros que se deben pasar son:\n'\
237   'Succion del motor (65% -100%)\n'\
      'Tiempo calentamiento en minutos\n'\
239   'Electrovalvula abrir\n')
      return 0
241
    ## Electrovalvula a abrir
243 valvula = float(sys.argv[3])
    if valvula <0 and valvula >5:
245   print ('PARAMETRO DE VALVULA INCORRECTO.\n' \
      'Los parametros que se deben pasar son:\n'\
247   'Succion del motor (65% -100%)\n'\
      'Numero de ciclos de calentamiento (>1)\n'\
249   'Electrovalvula abrir\n')
      return 0
251 valvula = int(valvula)
    ## Vector que contiene el orden de conmutacion de las electrovalvulas
253 ##vectorValvulas = numpy.random.randint(4,5,tiempo*60/cambio)
    ##vectorValvulas =[4,4,4]
255 print(succion)
    ## Se configura los puertos ADC, PWM
257 motorStart(succion)
    ADC.setup()
259 PWM.start(heatPin2600,95)


261
    samples = 0
263 while samples < 10:
      tick = time.time()
265   print ('Medidas de estabilizacion:\t'+str(readADC())) ## primera medida
      erronea
      samples+=1
267   tack = time.time();
```

```
    time.sleep(SLEEP-(tack-tick))

  vectorMin = numpy.random.randint(0,1,120)
  vectorMax = numpy.random.randint(100,101,60)
  ## creacion archivo de informacion experimento
  fileInfoTGS2600(valvula, succion,tiempo)

  ## Sistema de control
  print('Entro al primer bucle')
  contador = 0
  segundos = 1
  abrirElectrovalvula(valvula)

  while contador < 120:
    PWM.set_duty_cycle(heatPin2600,1)
    tick = time.time()
    controlSistema(segundos, 1)
    contador+=1
    segundos+=1
    tack = time.time();
    time.sleep(SLEEP-(tack-tick))

  print ('sALGO DEL PRIMER BUCLE')
  contador = 0

  while contador < tiempo*60:
    PWM.set_duty_cycle(heatPin2600,100)
    tick = time.time()
    controlSistema(segundos,100)
    contador+=1
    segundos+=1
    tack = time.time();
    time.sleep(SLEEP-(tack-tick))

  finExperimento()
  return 0

if __name__ == '__main__':
  try:
    main()
  except KeyboardInterrupt:
    print ('Interrupted')
    finExperimento()
    sys.exit(0)
```

resources/impulso.py

## I.5. Código para la modulación en lazo cerrado con un controlador PID

### I.5.1. Código para la primera versión de la placa de experimentación actualizando la señal de referencia basado en los valores mínimos y máximos de la señal

```
## principal_v05.2
## version que parte de principal_v05.1py en la que se realiza el control del
## cambio de minimos y maximos para controlar la adaptacion de la senal.
## sobre este script se la adaptacion del aire
import Adafruit_BBIO.ADC as ADC
```

```
import Adafruit_BBIO.GPIO as GPIO
import Adafruit_BBIO.PWM as PWM
import Adafruit_DHT
import sys
import os
import time
import math
import random
import numpy
from datetime import datetime, date
import _thread
from math import ceil

from  math import sin
from  math import pi

## Asignacion de puerto
sensorTemp22 = Adafruit_DHT.DHT22

Temp22       = 'P8_11' #Pin de lectura de temperatura/Humedad con DHT22 (Puerto GPIO_45)
electrovalve1 = 'P8_10' #Electrovalvula 1 (METANOL)
electrovalve2 = 'P8_12' #Electrovalvula 2 (ETANOL)
electrovalve3 = 'P8_14' #Electrovalvula 3 (BUTANOL)
electrovalve4 = 'P8_16' #Electrovalvula 4 (AIRE)
##electrovalve5 = 'P8_18' #Electrovalvula 5 (odoranteX)

motorPin     = 'P9_21' #Motor de succion
heatPin2600   = 'P9_22' #Calentamiento sensor TGS2600 (Puerto GPIO_2)
sensorPin2600 = 'P9_40' #Sensor TGS2600 (Puerto AIN_1)

## DECLARACION DE SALIDAS DEL SISTEMA.
GPIO.setup("P8_10",GPIO.OUT)
GPIO.setup("P8_12",GPIO.OUT)
GPIO.setup("P8_14",GPIO.OUT)
GPIO.setup("P8_16",GPIO.OUT)
##GPIO.setup("P8_18",GPIO.OUT)

## DECLARACION VARIABLES DEL SISTEMA

SLEEP = 1      ## Periodo de estabilizacion  para cada adquisicion
target = []      ## Almaceno los valores de la senal objetivo
periodo = 50       ## Periodo de la senal Target (segundos)
errorControl = 0.05   ## Parametro para controlar el tracking en %0.3
temperaturaMax = 100 ## Ciclo pwm maximo temperatura sensor
temperaturaMin = 1 ## Cilco pwm minimo temperatura sensor

tempMaxMaxControl = 95 ## Ciclo pwm maximo temperatura sensor para controlar la senal target
tempMaxMinControl = 85 ## Ciclo pwm maximo temperatura sensor para controlar la senal target

tempMinMaxControl = 15 ## Cilco pwm minimo temperatura sensor para controlar la senal target
tempMinMinControl = 5 ## Cilco pwm minimo temperatura sensor para controlar la senal target

muestrasTGS =[]   ## Almacena las muestras de odorantes
temperaturaTGS =[]  ## Almacena la temperatura de calentamiento  del sensor
rS_TGS = []      ## Almacena la resistencia del sensor
maximo = 0       ## Almaceno el valor maximo de la senal objetivo
minimo =0      ## Alamceno el valor minimo de la senal objetivo
```

```
63 numCiclos = 1   ## Variable para controlar el numero de ciclos calentamiento
       previo del sistema

65 tempActual =0
   lectura = 10;
67 alfa = 0.3

69 ## VARIABLES SENSOR
   Rl = 470  #omh
71 Vc = 5    #Voltios

73 SLEEP_tyh = 59  ##Medida de temperatura y humedad

75 ## VARIABLES PARA EL PID
   Kp = 0.6*alfa      ## Constante proporcional
77 Kd =  0.125*periodo##40.0/8          ## Constante derivativa
   Ki = (0.5)*periodo##40.0/2          ## Constante integral
79


81 ## Datos para el fichero de salida
   nombreArchivo = "pidTgs2600.txt"
83 archivoDatos = "pidTgs2600.dat"
   archivoTyH = "pidTgs2600TyH.data"
85 fileString = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime()) +'_'+
       nombreArchivo
   fileStringDatos = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime()) +'_'+
       archivoDatos
87 fileStringTyH = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime()) +'_'+
       archivoTyH
   actual = datetime.now()
89 ruta = "/media/microSD/PIDCONTROL/"+str(actual.month)+"/"+str(actual.day)+"/"
   if not os.path.exists(ruta): os.makedirs(ruta)
91
   rutaArchivo = ruta.strip()+str(fileString)
93 rutaArchivoDatos = ruta.strip()+str(fileStringDatos)
   rutaArchivoTyH = ruta.strip()+str(fileStringTyH)
95
   ## Se crean los ficheros de datos y temperatura
97 f = open(rutaArchivo,"w+")
   datos = open(rutaArchivoDatos,"w+")
99 ##tempH = open(rutaArchivoTyH,"w+")

101 def fileInfoTGS2600(vectorValvulas, succion, heat2600, tiempo, target,Kp,Ki,Kd,
        numCiclos,periodo):
      f = open(rutaArchivo,"w+")
103   f.write ('
        /*************************************************************************/'
        )
      f.write ('Plataforma de experimentacion BBB\n')
105   f.write ('Carlos Velasco\n')
      f.write ('Sensor TGS2600\n')
107   f.write ('-modluacion en lazo cerrado PID\n')
      fecha = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime())
109   f.write ('Fecha y hora de inicio: ' + str(fecha) + '\n')
      f.write ('Ruta del fichero: ' + str(ruta) + '\n')
111   f.write ('Nombre archivo: ' + str(fileString) + '\n')
      f.write ('Nombre archivo de datos: ' + str(fileStringDatos) + '\n')
113   f.write ('Nombre archivo de TyH: ' + str(fileStringTyH) + '\n')
      f.write ('Conmutacion entre electrovalvulas: ' +str(vectorValvulas)+ '\n\n')
115   f.write ('Parametros requeridos para el experimento \n')
      f.write ('Succion del motor (50-100) >>> '+ str(succion)+ '%\n')
117   f.write ('Temperatura Promedio TGS2600 (50-100) >>> '+str(heat2600)+'% Pin PWM
        : ' +str(heatPin2600)+ '\n\n')
```

```
    f.write ('Control Proporcional: '+str(Kp)+'\n')
    f.write ('Control Integral: '+str(Ki)+'\n')
    f.write ('Control Derivativo: '+str(Kd)+'\n')
    f.write ('Ciclos de precalentamiento: '+str(numCiclos)+'\n')
    f.write ('Periodo: '+str(periodo)+'\n')
    f.write ('Duracion del experimento: ' +str(tiempo)+ ' minutos\n')
    f.write ('
      /*****************************************************************\n'
      )
    f.write ('Target Inicial:\n')
    f.write (str(target)+'\n')
    f.write ('DATOS OBTENIDOS:\n\n')

def sensorTyH(tiempo):
    j=0
    tiempo = round(tiempo)
    time.sleep(9)
    # proceso de lectura
    while(j<tiempo):
        tickHT = time.time()
        humedad, temperatura = Adafruit_DHT.read_retry(sensorTemp22, Temp22)
        ahora = datetime.now()
        tackHT = time.time()
        ticktack = tackHT-tickHT

        if ticktack >SLEEP_tyh:
            print ("Tiempo medicion H y T > SLEEP:", ticktack)
        else:
            print ("Tiempo medicion H y T:",ticktack)
            if humedad is not None and temperatura is not None:
                print ('\nSensor DHT22: ' +str(sensorTemp22))
                print ('>>> '+str(ahora)+' Temp = ' +str(temperatura)+ ' Humidity = ' +
        str(humedad)+'\n')
                tempH = open(rutaArchivoTyH, "a")
                wline_h = str(ahora)+'\t'+str(temperatura)+'\t'+str(humedad)+'\n'
                tempH.writelines(wline_h)
                tempH.flush()
            else:
                print ('Failed to get reading, Try again!')
                tempH = open(rutaArchivoTyH, "a")
                wline_h=str(ahora)+' '+str(temperatura)+' '+str(humedad)+'\n'
                tempH.writelines(wline_h)
                tempH.flush()

        tack=time.time()
        p = tack - tickHT;
        time.sleep(SLEEP_tyh-p)
        j+=1

"""*****************************************************************
                        FUNCIONES DEL SISTEMA
*****************************************************************"""

def abrirElectrovalvula(electrovalvula):
    if electrovalvula ==1:
        print ('electrovalvula 1\n')
        GPIO.output(electrovalve1, GPIO.LOW)
        GPIO.output(electrovalve2, GPIO.HIGH)
        GPIO.output(electrovalve3, GPIO.HIGH)
        GPIO.output(electrovalve4, GPIO.HIGH)
    elif electrovalvula == 2:
        print ('electrovalvula 2\n')
        GPIO.output(electrovalve1, GPIO.HIGH)
```

```
    GPIO.output(electrovalve2, GPIO.LOW)
179     GPIO.output(electrovalve3, GPIO.HIGH)
    GPIO.output(electrovalve4, GPIO.HIGH)
181   elif electrovalvula == 3:
    print ('electrovalvula 3\n')
183     GPIO.output(electrovalve1, GPIO.HIGH)
    GPIO.output(electrovalve2, GPIO.HIGH)
185     GPIO.output(electrovalve3, GPIO.LOW)
    GPIO.output(electrovalve4, GPIO.HIGH)
187   elif electrovalvula == 4:
    print ('electrovalvula 4\n')
189     GPIO.output(electrovalve1, GPIO.HIGH)
    GPIO.output(electrovalve2, GPIO.HIGH)
191     GPIO.output(electrovalve3, GPIO.HIGH)
    GPIO.output(electrovalve4, GPIO.LOW)
193
def cerrarElectrovalvula():
195   GPIO.output(electrovalve1, GPIO.LOW)
  GPIO.output(electrovalve2, GPIO.LOW)
197   GPIO.output(electrovalve3, GPIO.LOW)
  GPIO.output(electrovalve4, GPIO.LOW)
199
def motorStart(succion):
201   PWM.start(motorPin,succion)

203 def readADC():
  valor = (1800*(ADC.read(sensorPin2600)))
205   valor = round(valor,3)
  return valor
207
def crearTarget(Vmax, Vmin, periodo):
209     global target
    print('maximo:\t'+str(Vmax)+'\t minimo:\t'+str(Vmin))
211     """ Asen(BX+C)"""
    A = (Vmax - Vmin)/2.0
213     B = (2*pi)/(periodo) ## periodo

215     senal =[]

217     for x in range(periodo+1): ## periodo+1
      senal.append(round(((A*sin(B*x))+(A+Vmin)),3))
219
    target = senal[0:periodo] ## senal[1:periodo]
221     print (target)

223 def esperar(tiempo):
  time.sleep(tiempo)
225
def mediaCalententamiento():
227   print ('Entro mediaCalententamiento')
  value = 0;
229   NM = 0;
  while NM<lectura:
231     tick = time.time()
    adcValue = readADC()
233     print ('adcValue: \t'+str(adcValue))
    value += adcValue
235     NM+=1
    tack = time.time();
237     time.sleep(SLEEP-(tack-tick))
  value/=lectura
239   value = round(value)
  print ('Media:\t'+str(value))
```

```
241   return value

243 def generarTarget(temperatura,periodo,max,min):
    global maximo
245   global minimo
    ##ADC.read_raw(sensorPin2600)  ## primera medida erronea
247   abrirElectrovalvula(4) ## se abre electrovalvula del aire
    ## Se toman muestras para el valor maximo 80% (4V)
249  PWM.set_duty_cycle(heatPin2600, temperaturaMax);
    esperar(1)
251   ##ADC.read_raw(sensorPin2600)  ## primera medida erronea
    ##maximo = ADC.read_raw(sensorPin2600)
253   ##maximo = mediaCalententamiento()

255   ## Se toman muestras para el valor minimo 10% (1V)
    PWM.set_duty_cycle(heatPin2600, temperaturaMin);
257   esperar(1)
    ##ADC.read_raw(sensorPin2600)  ## primera medida erronea
259   ##minimo = ADC.read_raw(sensorPin2600)
    ##minimo = mediaCalententamiento()
261
    ## Escritua por pantalla de valores obtenidos
263   print ('Valor maximo de lectura: '+str(maximo)+ 'V')
    print ('Valor minimo de lectura: '+str(minimo)+ 'V')
265   crearTarget(max, min, periodo)

267 def controlSistema2(temperatura):
    ##ADC.read_raw(sensorPin2600)  ## primera medida erronea
269   value = readADC()
    if(value<1):
271     value=1
    ##print ('value:\t'+str(value))
273   captura = datetime.now();

275   ## Caclulo del valor de resistencia interna del sensor
    ## el valor devuelto por read_raw ata en mV por eso divide entre 1000
277   RStgs2600 = ((Vc*Rl)/(value/1000.0)) - Rl


279



281   ## Se muestra por pantalla los datos
    ##print ('Muestra PID['+str(0)+']\tTarget: '+str(value)+'\tValor:'+str(value)+'
      mV\tRs: '+str(RStgs2600)+'\tTemperatura: '+str(temperatura)+'\t'+str(captura)
      +'\n')
283
    ## Se guardan datos en archivo
285   ##f = open(rutaArchivo,"a")
    ##datos = open(rutaArchivoDatos,"a")
287   ##wline_datos = str(0)+'\t'+str(value)+'\t'+str(value)+'\t'+str(RStgs2600)+'\t
      '+str(temperatura)+'\t'+str(temperatura)+'\t'+str(captura)+'\n'
    ##wline_f = 'Muestra PID['+str(0)+']\tTarget: '+str(value)+'\tValor:'+str(value
      )+'mV\tRs: '+str(RStgs2600)+'\tTemperatura: '+str(temperatura)+'\t'+str(
      captura)+'\n'
289   ##datos.writelines(wline_datos)
    ##datos.flush()
291   ##f.writelines(wline_f)
    ##f.flush()
293
  def calentamientoPrevio(vectorT,vectorTd,valvula):
295   contador = 0
    ciclo = 0
297   abrirElectrovalvula(valvula)
    while ciclo < numCiclos:
```

```
299     PWM.set_duty_cycle(heatPin2600,vectorTd[0])
        while contador<20:
301       tick = time.time()
          ##print("Temperatura:\t"+str(vectorTd[0]))
303       controlSistema2(vectorTd[0])
          contador+=1
305       tack = time.time();
          time.sleep(SLEEP-(tack-tick))
307
        contador=0
309     while contador < len(vectorTd):
          PWM.set_duty_cycle(heatPin2600,vectorTd[contador])
311       tick = time.time()
          ##print("Temperatura:\t"+str(vectorTd[contador]))
313       controlSistema2(vectorTd[contador])
          contador+=1
315       tack = time.time();
          time.sleep(SLEEP-(tack-tick))
317
        contador=0
319     PWM.set_duty_cycle(heatPin2600,vectorTd[19])
        while contador<20:
321       tick = time.time()
          ##print("Temperatura:\t"+str(vectorTd[19]))
323       controlSistema2(vectorTd[19])
          contador+=1
325       tack = time.time();
          time.sleep(SLEEP-(tack-tick))
327
        contador=0
329     while contador < len(vectorT):
          PWM.set_duty_cycle(heatPin2600,vectorT[contador])
331       tick = time.time()
          ##print("Temperatura:\t"+str(vectorT[contador]))
333       controlSistema2(vectorT[contador])
          contador+=1
335       tack = time.time();
          time.sleep(SLEEP-(tack-tick))
337
        contador=0
339     ciclo+=1

341 def pidController(Kp, Ki, kd,SetPoint,SensorValue, lastError,lastTime, addError):
    """
343     deltaU =Kp*(ep + Kd*ed + (1/Ki)*ei)
    """
345   sampleTime = 1;

347   error = round((SetPoint - SensorValue),3) ## cambio
    currentTime = time.time()
349   deltaTime = 1.0##currentTime - lastTime
    deltaError = error - lastError
351

353   ep = error
    ed = (deltaError)/(deltaTime)
355   ei = addError + (error*deltaTime)

357   output = Kp *(ep +(Kd*ed) + ((1/Ki)*ei))

359   datos = [output,error,ei,currentTime]
    print(datos)
361   return datos
```

```
363 """Funcion que se encarga de llevar a acabo el experimento"""
    def controlSistema(contador, subtarget,muestra,lastError,lastTime,addError,temp,
        flagTarget):
365   ##ADC.read_raw(sensorPin2600)  ## primera medida erronea
      global maximo
367   global minimo
      global temPIDmax
369   global temPIDmin

371   flagMaximo = False;  ## para cuando es el maximo
      flagMinimo = False;  ## para cuando es el minimo
373

375   value = readADC()
      if(value<1):
377     value=1
      print ('value:\t'+str(value))
379   captura = datetime.now();

381   print ("LasError:"+str(lastError)+"\t lastTime:"+str(lastTime)+"\t addError:"+
        str(addError)+"\t Temperatura:"+str(temp)+"\t FlagTarget:"+str(flagTarget))

383   ##value /=1000.0
      ## actualizo los valores maximos y minimos de la senal
385   if subtarget == max(target):
        flagMaximo  = True;
387     print ('Entro al maximo '+ str(flagMaximo))
        e = subtarget*errorControl;
389     print ('error:\t'+str(e))
        if not(value>=(subtarget-e) and value<=(subtarget+e)):
391       print ('El maximo cambia')
          maximo = value
393     else:
          maximo = max(target)
395   if subtarget == min(target):
        flagMinimo= True
397     print ('Entro al minimo ' +str(flagMinimo))
        e = subtarget*errorControl;
399     print('error:\t'+str(e))
        if not(value>=(subtarget-e) and value<=(subtarget+e)):
401       print ('El minimo cambia')
          minimo = value
403     else:
          minimo = min(target)
405


407   ## creo el parametro pid para llamar a la debida funcion
      print ('sutarget:\t'+str(subtarget))
409
      datosOut = pidController(Kp, Ki, Kd,subtarget,value, lastError,lastTime,
        addError)
411   ##pid.SetPoint = subtarget

413   ##pid.update(value)
      errorMax = max(target)
415   ##print("Error Maximo: "+str(errorMax))
      errorMin =0;
417   ##print("Error Minimo: "+str(errorMin))


419
      """print(str(pid.Kp)+"\t"+str(pid.Ki)+"\t"+str(pid.Kd)+"\n"\
421   +str(pid.sample_time)+"\t"+str(pid.current_time)+"\t"+str(pid.last_time)+"\n"\
```

```
    +str(pid.last_error)+"\n"
423 +str(pid.PTerm)+"\t"+str(pid.ITerm)+"\t"+str(pid.DTerm));"""

425 Vout = datosOut[0]##pid.output/1000.0
    error = datosOut[1]
427 print("pid (mv):\t"+str(datosOut[0])+" error:\t"+str(error))

429 if (error >=-0.5 and error <= 0.5):##cambio error ==0
      t1 = 0##temp
431 ##elif Vout>0:
    else:
433   t1 = (Vout-errorMax)*((temperaturaMin-temperaturaMax)/(errorMin-errorMax))+
      temperaturaMax
      ##temperaturaPID = (Vout*100.0)/5.0
435 """else:
      Vp = ((Vout)*-1)+errorMax;
437   t1 = abs(((Vp-errorMax)*((temperaturaMin-temperaturaMax)/(errorMin-errorMax))
      +temperaturaMax)-temp)"""

439 t1 = round(t1,2)
    print ('temperatura PID:\t'+str(t1))
441
    """if((temp == temperaturaMax and Vout ==0)):
443   temperaturaPID = temp-t1
    elif (Vout >= 1 and Vout <= 2):
445   temperaturaPID = temp-t1
    else:
447   temperaturaPID = temp+t1"""

449 if(temp == temperaturaMax and error==0):
      temperaturaPID = temp
451 elif(temp == temperaturaMax and error>0):
      temperaturaPID = temp
453 elif(temp == temperaturaMax and error<0):
      if(t1>0):
455     temperaturaPID = temp - t1
      else:
457     temperaturaPID = temp + t1

459 elif(temp < temperaturaMax and error==0):
      temperaturaPID = temp
461 elif(temp < temperaturaMax and error>0):
      temperaturaPID = temp +t1
463 elif(temp < temperaturaMax and error<0):
      if(t1>0):
465     temperaturaPID = temp - t1
      else:
467     temperaturaPID = temp + t1

469 elif(temp > temperaturaMax and  (error==0 or error > 0)):
      temperaturaPID = temp
471 elif(temp > temperaturaMax and error<0):
      temperaturaPID = temp + t1
473
    """elif (Vout >= 1 and Vout <= 3):
475   temperaturaPID = temp - (temp*(0.01))"""

477 temperaturaPID = round(temperaturaPID,2)

479 if temperaturaPID > temperaturaMax:
      print ('temperatura PID mayor a max:\t'+str(temperaturaPID))
481   temperaturaPID = temperaturaMax
    elif temperaturaPID < temperaturaMin:
```

```
483     print ('temperatura PID menor a min:\t'+str(temperaturaPID))
        temperaturaPID = temperaturaMin
485
    datosOut.append(temperaturaPID)
487 ## actualizo la temperatura del sensor
    PWM.set_duty_cycle(heatPin2600, temperaturaPID);
489


491 muestrasTGS.append(value)
    temperaturaTGS.append(temperaturaPID)
493
    ## Caclulo del valor de resistencia interna del sensor
495 ## el valor devuelto por read_raw ata en mV por eso divide entre 1000
    RStgs2600 = ((Vc*Rl)/(value/1000.0)) - Rl
497 RStgs2600 = round(RStgs2600,2)
    ##rS_TGS.append(tgs2600RS)
499
    ## Se muestra por pantalla los datos
501 print ('Muestra PID['+str(contador)+']\tTarget: '+str(subtarget)+'\tValor:'+str
      (value)+'V\tRs: '+str(RStgs2600)+'\tTemperatura: '+str(temp)+'\tTemperatura: '
      +str(temperaturaPID)+'\t'+str(captura)+'\n')

503 ## Se guardan datos en archivo
    f = open(rutaArchivo,"a")
505 datos = open(rutaArchivoDatos,"a")
    wline_datos = str(contador)+'\t'+str(subtarget)+'\t'+str(value)+'\t'+str(
      RStgs2600)+'\t'+str(temp)+'\t'+str(temperaturaPID)+'\t'+str(captura)+'\n'
507 wline_f = 'Muestra PID['+str(contador)+']\n>>Target: '+str(subtarget)+'V >>
      Valor:'+str(value)+'V >> Rs: '+str(RStgs2600)+' Temperatura: '+str(temp)+' '+'
       TemperaturaPID: '+str(temperaturaPID)+' '+str(captura)+'\n'
    datos.writelines(wline_datos)
509 datos.flush()
    f.writelines(wline_f)
511 f.flush()

513 if(flagMaximo):
      temPIDmax = temperaturaPID
515   print("Temperatura del maximo: " +str(temPIDmax)+"\n")

517 if (flagMinimo):
      temPIDmin = temperaturaPID
519   print("Temperatura del minimo: " +str(temPIDmin)+"\n")


521
    ## se calcula el nuevo target cada vez que se termine el periodo
523 if(flagTarget):

525   print('Valores previos: maximo:\t'+str(maximo)+'\t minimo:\t'+str(minimo))
      print('Valores previos: Tmax:\t'+str(temPIDmax)+'\t Tmin:\t'+str(temPIDmin))
527   print('max(target)- maximo =\t'+str(max(target)- maximo))
      print('maximo - minimo =\t'+str(maximo- minimo))
529   ## guardo los valores temporales del maximo y minimo
      maxAux = 0.0
531   minAux = 0.0
      ##primero compruebo que la amplitud de la senal sea superior a un valor
533   if((maximo-minimo)<150.0):

535     if(maximo <= minimo):
          maxAux =  minimo
537       minAux = minimo - (minimo*0.7)
        else:
539       ## El maximo no llega a ser cercano al maximo del target
          if((max(target)- maximo)>=0):
```

```
            if(temPIDmax > tempMaxMaxControl):
              maxAux = ceil(maximo-(maximo*0.1))
            else:
              maxAux = ceil(maximo)
          else:
            if(temPIDmax > tempMaxMaxControl):
              maxAux = ceil(maximo-(maximo*0.1))
            else:
              maxAux = maximo

          ## Se define el valor del minimo
          if((min(target)-minimo)<=0):
            if (temPIDmin > tempMinMaxControl):
              minAux = ceil(maxAux -(maxAux*0.9))
            else:
              minAux = ceil (maxAux -(maxAux*0.5))
          else:
            if (temPIDmin > tempMinMaxControl):
              minAux = ceil(maxAux -(maxAux*0.9))
            else:
              minAux = ceil(maxAux -(maxAux*0.5))
      else:
        if (temPIDmin > tempMinMaxControl):
          minAux = min(target)-(min(target)*0.05) ##minimo - (minimo*0.1)
        elif (temPIDmin < tempMinMinControl):
          minAux = min(target)+(min(target)*0.05) ##minimo + (minimo*0.1)
        else:
          minAux = minimo


        if((max(target)- maximo)>0):
          maxAux = ceil(maximo)
          ##if(temPIDmax < tempMaxMinControl):
          ##  maximo = maximo + (maximo*0.05)
          ##else:
          ##  maximo = maximo
        else:
          if(temPIDmax > tempMaxMaxControl):
            maxAux = ceil(max(target) -max(target)*0.02)
          elif (temPIDmax < tempMaxMinControl):
            maxAux = ceil(max(target) +max(target)*0.02)

        ##if(temPIDmin == temPIDmax):
        ##  maximo = ceil(maximo +(maximo*0.3))
        ##  minimo = minimo -(minimo*0.3)

      if(maxAux >1800.0):
        maxAux = 1800.0

      if(minAux < 0.0):
        minAux = 1800.0*0.01



      crearTarget(maxAux,minAux,periodo)
      flagMaximo = False
      flagMinimo = False

    return datosOut

def finExperimento():
  PWM.stop(heatPin2600)
  PWM.stop(motorPin)
```

```
    PWM.cleanup()
605 ##_thread.exit()
    cerrarElectrovalvula()
607
    fecha_fin=datetime.now()
609 print('\nEXPERIMENTO FINALIZADO CON EXITO')
    print ('Experimento terminado: '+str(fecha_fin))
611 f = open(rutaArchivo, "a")
    f.write('\nEXPERIMENTO FINALIZADO CON EXITO\n')
613 f.write('Fecha y hora de fin de experimentacion: ' +str(fecha_fin))
    f.flush()
615 f.close()
    datos.close()
617 ##tempH.close()

619 def main():
    ## varaibles de control global
621 global temPIDmax
    global temPIDmin
623 lastError =0;
    addError =0;
625 datos =[]
    print ('************************************************************\n'\
627 '         ALGORITMO DE MODULACION EN LAZO CERRADO CONTROL PID \n'\
    '************************************************************\n' )
629
    if(len(sys.argv)<8):
631   print ('\n NUMERO DE PARAMETROS INCORRECTOS\n' \
      'Los parametros que se deben pasar son:\n'\
633   'Succion del motor (65% -100%)\n'
      'Temperatura inicial TGS2600 (50%- 100%)\n'\
635   'Tiempo de experimentacion (minutos)\n'\
      'Tiempo de conmutacion  electrovalvulas (segundos)\n')
637   return 0

639 ## Succion del motor 50-100%
    succion = float(sys.argv[1])
641 if succion>100 or succion<65:
      print ('PARAMETRO DE SUCCION DE MOTOR INCORRECTO.\n' \
643   'Los parametros que se deben pasar son:\n'\
      'Succion del motor (65% -100%)\n'
645   'Temperatura inicial TGS2600 (50%- 100%)\n'\
      'Tiempo de experimentacion (minutos)\n'\
647   'Tiempo de conmutacion  electrovalvulas (segundos)\n')
      return 0
649
    ## Temperatura media del sensor TGS2600
651 heat2600 = float(sys.argv[2])
    if heat2600 >100 or heat2600 <0:
653   print ('PARAMETRO DE CALENTAMIENTO DEL SENSOR  INCORRECTO.\n' \
      'Los parametros que se deben pasar son:\n'\
655   'Succion del motor (50% -100%)\n'
      'Temperatura inicial TGS2600 (50%- 100%)\n'\
657   'Tiempo de experimentacion (minutos)\n'\
      'Tiempo de conmutacion  electrovalvulas (segundos)\n')
659   return 0

661 ## Duracion del experimento
    tiempo = float(sys.argv[3])
663 if tiempo <1:
      print ('EL TIEMPO DE EXPERIMENTACION ES INCORECTO.\n' \
665   'Los parametros que se deben pasar son:\n'\
      'Succion del motor (50% -100%)\n'
```

```
667     'Temperatura inicial TGS2600 (50%- 100%)\n'\
        'Tiempo de experimentacion (minutos)\n'\
669     'Tiempo de conmutacion  electrovalvulas (segundos)\n')
        return 0
671   ## Tiempo conmutacion electrovalvula
      cambio = float(sys.argv[4])
673   if cambio <1:
        print ('EL TIEMPO DE CONMUTACION DE VALVULAS ES INCORRECTO.\n' \
675     'Los parametros que se deben pasar son:\n'\
        'Succion del motor (50% -100%)\n'
677     'Temperatura inicial TGS2600 (50%- 100%)\n'\
        'Tiempo de experimentacion (minutos)\n'\
679     'Tiempo de conmutacion  electrovalvulas (segundos)\n')
        return 0
681
      odorante = float(sys.argv[5])
683   if odorante <1 and odorante>5:
        print ('LA ELECTROVALVULA SELECCIONADA NO ES CORRECTA.\n' \
685     'Los parametros que se deben pasar son:\n'\
        'Succion del motor (50% -100%)\n'
687     'Temperatura inicial TGS2600 (50%- 100%)\n'\
        'Tiempo de experimentacion (minutos)\n'\
689     'Tiempo de conmutacion  electrovalvulas (segundos)\n'\
        'Electrovalvula incorrecta, valor entre 1 y 4\n')
691     return 0

693   pico = float(sys.argv[6])
      if pico<0:
695     print ('VALOR MAXIMO INCORRECTO.\n' \
        'Los parametros que se deben pasar son:\n'\
697     'Succion del motor (50% -100%)\n'
        'Temperatura inicial TGS2600 (50%- 100%)\n'\
699     'Tiempo de experimentacion (minutos)\n'\
        'Tiempo de conmutacion  electrovalvulas (segundos)\n'\
701     'Electrovalvula incorrecta, valor entre 1 y 4\n')
        return 0
703
      bajo = float(sys.argv[7])
705   if bajo<0:
        print ('VALOR MAXIMO INCORRECTO.\n' \
707     'Los parametros que se deben pasar son:\n'\
        'Succion del motor (50% -100%)\n'
709     'Temperatura inicial TGS2600 (50%- 100%)\n'\
        'Tiempo de experimentacion (minutos)\n'\
711     'Tiempo de conmutacion  electrovalvulas (segundos)\n'\
        'Electrovalvula incorrecta, valor entre 1 y 4\n')
713     return 0


715


717   ## Vector que contiene el orden de conmutacion de las electrovalvulas
      vectorValvulas = [1, 1, 2, 2, 3, 3, 4, 4, 2, 1, 3, 1, 4, 3, 2, 4, 1]#
        #[1,2,3]##[int(odorante)]##
719   ##numpy.random.randint(1,5,int((tiempo*60)/cambio)) ##[int(odorante)] ## numpy.
        random.randint(1,5,int((tiempo*60)/cambio))

721   ##vectorValvulas =[4,4,4]

723   vectorT= range(0,100,5)
      vectorTd= range(95,-5,-5)
725
      ## Se configura los puertos ADC, PWM
727   motorStart(succion)
```

```
ADC.setup()
PWM.start(heatPin2600,heat2600)


samples = 0
while samples < 30:
  tick = time.time()
  print ('Medidas de estabilizacion:\t'+str(readADC())) ## primera medida
  erronea
  samples+=1
  tack = time.time();
  time.sleep(SLEEP-(tack-tick))

## precalentamiento del sensor
calentamientoPrevio(vectorT,vectorTd,int(odorante))

## Generar Target
generarTarget(heat2600, periodo,pico,bajo)
print(str(heat2600))
PWM.set_duty_cycle(heatPin2600, heat2600);
esperar(3)

## Llamada al hilo para la medida de temperatura y humedad
##_thread.start_new_thread(sensorTyH,(tiempo,))

## creacion archivo de informacion experimento
fileInfoTGS2600(vectorValvulas, succion, heat2600, tiempo, target,Kp,Ki,Kd,
  numCiclos,periodo)

## Sistema de control
contador = 1
muestra = 0;
pos = 0 ;   ## cambiar este valor solo es una prueba "aire"
temp = heat2600;
flagTarget = False
temPIDmax =0.0
temPIDmin =0.0
"""print(str(pid.Kp)+"\t"+str(pid.Ki)+"\t"+str(pid.Kd)+"\n"\
+str(pid.sample_time)+"\t"+str(pid.current_time)+"\t"+str(pid.last_time)+"\n"\
+str(pid.last_error)+"\n"
+str(pid.PTerm)+"\t"+str(pid.ITerm)+"\t"+str(pid.DTerm));
tempActual = heat2600"""
lastTime = time.time()
while contador < tiempo*60:

  electrovalvula = vectorValvulas[pos]
  if electrovalvula == 1:
    print ('ELECTROVALVULA: '+str(electrovalvula)+' METANOL\n')
  elif electrovalvula == 2:
    print ('ELECTROVALVULA: '+str(electrovalvula)+' ETANOL \n')
  elif electrovalvula == 3:
    print ('ELECTROVALVULA: '+str(electrovalvula)+' BUTANOL \n')
  else:
    print ('ELECTROVALVULA: '+str(electrovalvula)+' AIRE \n')
  abrirElectrovalvula(electrovalvula)

  j = 0;
  while j < cambio:
    tick = time.time()
    if muestra == 0:
      subtarget = target[muestra]
      muestra+=1
      flagTarget = False
```

```
789         elif (muestra%(periodo-1) ==0):
              subtarget = target[periodo-1]
791           muestra = 0
              flagTarget = True
793         else:
              subtarget = target[muestra]
795           muestra+=1
              flagTarget = False
797
799
801         datos = controlSistema(contador, subtarget,muestra,lastError,lastTime,
        addError,temp,flagTarget)
            print ("temPIDmax:" +str(temPIDmax)+"\t temPIDmin:"+str(temPIDmin)+"\n")
803         lastTime = datos[3]
            addError = datos[2]
805         lastError = datos[1]
            temp = datos[4]
807         contador+=1
            j+=1
809         tack = time.time();
            time.sleep(SLEEP-(tack-tick))
811       pos+=1

813     finExperimento()
        return 0
815
    if __name__ == '__main__':
817     try:
          main()
819     except KeyboardInterrupt:
          print ('Interrupted')
821       finExperimento()
          sys.exit(0)
```

resources/principal_v05.2.py

### I.5.2. Código para la primera versión de la placa de experimentación del controlador PID

```
1 ## principal_v05.5
  ## version que parte de principal_v05.4.py en la que se reliza
3 ## Cambio del valor maximo de temeperatura que se aplicara al sensor
  ## cambio en los valores  para cuando el aire esta en proceso
5 ## de succion
  import Adafruit_BBIO.ADC as ADC
7 import Adafruit_BBIO.GPIO as GPIO
  import Adafruit_BBIO.PWM as PWM
9 import Adafruit_DHT
  import sys
11 import os
  import time
13 import math
  import random
15 import numpy
  from datetime import datetime, date
17 import _thread
  from math import ceil
19
  from  math import sin
```

```
21 from  math import pi

23 ## Asignacion de puerto
   sensorTemp22 = Adafruit_DHT.DHT22
25
   Temp22      = 'P8_11' #Pin de lectura de temperatura/Humedad con DHT22 (Puerto
       GPIO_45)
27 electrovalve1 = 'P8_10' #Electrovalvula 1 (METANOL)
   electrovalve2 = 'P8_12' #Electrovalvula 2 (ETANOL)
29 electrovalve3 = 'P8_14' #Electrovalvula 3 (BUTANOL)
   electrovalve4 = 'P8_16' #Electrovalvula 4 (AIRE)
31 ##electrovalve5 = 'P8_18' #Electrovalvula 5 (odoranteX)

33 motorPin     = 'P9_21' #Motor de succion
   heatPin2600   = 'P9_22' #Calentamiento sensor TGS2600 (Puerto GPIO_2)
35 sensorPin2600 = 'P9_40' #Sensor TGS2600 (Puerto AIN_1)

37 ## DECLARACION DE SALIDAS DEL SISTEMA.
   GPIO.setup("P8_10",GPIO.OUT)
39 GPIO.setup("P8_12",GPIO.OUT)
   GPIO.setup("P8_14",GPIO.OUT)
41 GPIO.setup("P8_16",GPIO.OUT)
   ##GPIO.setup("P8_18",GPIO.OUT)
43
   ## DECLARACION VARIABLES DEL SISTEMA
45
   SLEEP = 1      ## Periodo de estabilizacion  para cada adquisicion
47 target = []      ## Almaceno los valores de la senal objetivo
   periodo = 50       ## Periodo de la senal Target (segundos)
49 errorControl = 0.05   ## Parametro para controlar el tracking en %0.3
   temperaturaMax = 95 ## Ciclo pwm maximo temperatura sensor
51 temperaturaMin = 5 ## Cilco pwm minimo temperatura sensor

53 tempMaxMaxControl = 95 ## Ciclo pwm maximo temperatura sensor para controlar la
       senal target
   tempMaxMinControl = 85 ## Ciclo pwm maximo temperatura sensor para controlar la
       senal target
55
   tempMinMaxControl = 15 ## Cilco pwm minimo temperatura sensor para controlar la
       senal target
57 tempMinMinControl = 5 ## Cilco pwm minimo temperatura sensor para controlar la
       senal target

59 muestrasTGS =[]   ## Almacena las muestras de odorantes
   temperaturaTGS =[]  ## Almacena la temperatura de calentamiento  del sensor
61 rS_TGS = []      ## Almacena la resistencia del sensor
   maximo = 0       ## Almaceno el valor maximo de la senal objetivo
63 minimo =0     ## Alamceno el valor minimo de la senal objetivo
   numCiclos = 1   ## Variable para controlar el numero de ciclos calentamiento
       previo del sistema
65
   tempActual =0
67 lectura = 10;
   alfa = 0.3
69
   ## VARIABLES SENSOR
71 Rl = 470  #omh
   Vc = 5    #Voltios
73
   SLEEP_tyh = 59  ##Medida de temperatura y humedad
75
   ## VARIABLES PARA EL PID
77 Kp = 0.6*alfa      ## Constante proporcional
```

```
Kd =  0.125*periodo##40.0/8          ## Constante derivativa
Ki = (0.5)*periodo##40.0/2           ## Constante integral


## Datos para el fichero de salida
nombreArchivo = "pidTgs2600.txt"
archivoDatos = "pidTgs2600.dat"
archivoTyH = "pidTgs2600TyH.data"
fileString = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime()) +'_'+
    nombreArchivo
fileStringDatos = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime()) +'_'+
    archivoDatos
fileStringTyH = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime()) +'_'+
    archivoTyH
actual = datetime.now()
ruta = "/media/microSD/PIDCONTROL/"+str(actual.month)+"/"+str(actual.day)+"/"
if not os.path.exists(ruta): os.makedirs(ruta)

rutaArchivo = ruta.strip()+str(fileString)
rutaArchivoDatos = ruta.strip()+str(fileStringDatos)
rutaArchivoTyH = ruta.strip()+str(fileStringTyH)

## Se crean los ficheros de datos y temperatura
f = open(rutaArchivo,"w+")
datos = open(rutaArchivoDatos,"w+")
##tempH = open(rutaArchivoTyH,"w+")

def fileInfoTGS2600(vectorValvulas, succion, heat2600, tiempo, target,Kp,Ki,Kd,
    numCiclos,periodo):
  f = open(rutaArchivo,"w+")
  f.write ('/*****************************************************************/'
    )
  f.write ('Plataforma de experimentacion BBB\n')
  f.write ('Carlos Velasco\n')
  f.write ('Sensor TGS2600\n')
  f.write ('-modluacion en lazo cerrado PID\n')
  fecha = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime())
  f.write ('Fecha y hora de inicio: ' + str(fecha) + '\n')
  f.write ('Ruta del fichero: ' + str(ruta) + '\n')
  f.write ('Nombre archivo: ' + str(fileString) + '\n')
  f.write ('Nombre archivo de datos: ' + str(fileStringDatos) + '\n')
  f.write ('Nombre archivo de TyH: ' + str(fileStringTyH) + '\n')
  f.write ('Conmutacion entre electrovalvulas: ' +str(vectorValvulas)+ '\n\n')
  f.write ('Parametros requeridos para el experimento \n')
  f.write ('Succion del motor (50-100) >>> '+ str(succion)+ '%\n')
  f.write ('Temperatura Promedio TGS2600 (50-100) >>> '+str(heat2600)+'% Pin PWM
    : ' +str(heatPin2600)+ '\n\n')
  f.write ('Control Proporcional: '+str(Kp)+'\n')
  f.write ('Control Integral: '+str(Ki)+'\n')
  f.write ('Control Derivativo: '+str(Kd)+'\n')
  f.write ('Ciclos de precalentamiento: '+str(numCiclos)+'\n')
  f.write ('Periodo: '+str(periodo)+'\n')
  f.write ('Duracion del experimento: ' +str(tiempo)+ ' minutos\n')
  f.write ('/*****************************************************************/\n'
    )
  f.write ('Target Inicial:\n')
  f.write (str(target)+'\n')
  f.write ('DATOS OBTENIDOS:\n\n')

def sensorTyH(tiempo):
  j=0
  tiempo = round(tiempo)
  time.sleep(9)
```

```
  # proceso de lectura
135 while(j<tiempo):
    tickHT = time.time()
137 humedad, temperatura = Adafruit_DHT.read_retry(sensorTemp22, Temp22)
    ahora = datetime.now()
139 tackHT = time.time()
    ticktack = tackHT-tickHT
141
    if ticktack >SLEEP_tyh:
143   print ("Tiempo medicion H y T > SLEEP:", ticktack)
    else:
145   print ("Tiempo medicion H y T:",ticktack)
      if humedad is not None and temperatura is not None:
147     print ('\nSensor DHT22: ' +str(sensorTemp22))
        print ('>>> '+str(ahora)+' Temp = ' +str(temperatura)+ ' Humidity = ' +
    str(humedad)+'\n')
149     tempH = open(rutaArchivoTyH, "a")
        wline_h = str(ahora)+'\t'+str(temperatura)+'\t'+str(humedad)+'\n'
151     tempH.writelines(wline_h)
        tempH.flush()
153   else:
        print ('Failed to get reading, Try again!')
155     tempH = open(rutaArchivoTyH, "a")
        wline_h=str(ahora)+' '+str(temperatura)+' '+str(humedad)+'\n'
157     tempH.writelines(wline_h)
        tempH.flush()
159
    tack=time.time()
161 p = tack - tickHT;
    time.sleep(SLEEP_tyh-p)
163 j+=1

165 """*****************************************************************
                        FUNCIONES DEL SISTEMA
167 *****************************************************************"""

169 def abrirElectrovalvula(electrovalvula):
  if electrovalvula ==1:
171   print ('electrovalvula 1\n')
    GPIO.output(electrovalve1, GPIO.LOW)
173 GPIO.output(electrovalve2, GPIO.HIGH)
    GPIO.output(electrovalve3, GPIO.HIGH)
175 GPIO.output(electrovalve4, GPIO.HIGH)
  elif electrovalvula == 2:
177   print ('electrovalvula 2\n')
    GPIO.output(electrovalve1, GPIO.HIGH)
179 GPIO.output(electrovalve2, GPIO.LOW)
    GPIO.output(electrovalve3, GPIO.HIGH)
181 GPIO.output(electrovalve4, GPIO.HIGH)
  elif electrovalvula == 3:
183   print ('electrovalvula 3\n')
    GPIO.output(electrovalve1, GPIO.HIGH)
185 GPIO.output(electrovalve2, GPIO.HIGH)
    GPIO.output(electrovalve3, GPIO.LOW)
187 GPIO.output(electrovalve4, GPIO.HIGH)
  elif electrovalvula == 4:
189   print ('electrovalvula 4\n')
    GPIO.output(electrovalve1, GPIO.HIGH)
191 GPIO.output(electrovalve2, GPIO.HIGH)
    GPIO.output(electrovalve3, GPIO.HIGH)
193 GPIO.output(electrovalve4, GPIO.LOW)

195 def cerrarElectrovalvula():
```

```
    GPIO.output(electrovalve1, GPIO.LOW)
197 GPIO.output(electrovalve2, GPIO.LOW)
    GPIO.output(electrovalve3, GPIO.LOW)
199 GPIO.output(electrovalve4, GPIO.LOW)

201 def motorStart(succion):
    PWM.start(motorPin,succion)
203
    def readADC():
205   valor = (1800*(ADC.read(sensorPin2600)))
      valor = round(valor,3)
207   return valor

209 def crearTarget(Vmax, Vmin, periodo):
        global target
211     print('maximo:\t'+str(Vmax)+'\t minimo:\t'+str(Vmin))
        """ Asen(BX+C)"""
213     A = (Vmax - Vmin)/2.0
        B = (2*pi)/(periodo) ## periodo
215
        senal =[]
217
        for x in range(periodo+1): ## periodo+1
219       senal.append(round(((A*sin(B*x))+(A+Vmin)),3))

221     target = senal[0:periodo] ## senal[1:periodo]
        print (target)
223
    def esperar(tiempo):
225   time.sleep(tiempo)

227 def mediaCalententamiento():
      print ('Entro mediaCalententamiento')
229   value = 0;
    NM = 0;
231   while NM<lectura:
        tick = time.time()
233     adcValue = readADC()
        print ('adcValue: \t'+str(adcValue))
235     value += adcValue
      NM+=1
237     tack = time.time();
        time.sleep(SLEEP-(tack-tick))
239   value/=lectura
      value = round(value)
241   print ('Media:\t'+str(value))
      return value
243
    def generarTarget(temperatura,periodo,max,min):
245   global maximo
      global minimo
247   ##ADC.read_raw(sensorPin2600) ## primera medida erronea
      abrirElectrovalvula(4) ## se abre electrovalvula del aire
249   ## Se toman muestras para el valor maximo 80% (4V)
    PWM.set_duty_cycle(heatPin2600, temperaturaMax);
251   esperar(1)
      ##ADC.read_raw(sensorPin2600) ## primera medida erronea
253   ##maximo = ADC.read_raw(sensorPin2600)
      ##maximo = mediaCalententamiento()
255
      ## Se toman muestras para el valor minimo 10% (1V)
257   PWM.set_duty_cycle(heatPin2600, temperaturaMin);
      esperar(1)
```

```
259   ##ADC.read_raw(sensorPin2600)  ## primera medida erronea
      ##minimo = ADC.read_raw(sensorPin2600)
261   ##minimo = mediaCalententamiento()

263   ## Escritua por pantalla de valores obtenidos
      print ('Valor maximo de lectura: '+str(maximo)+ 'V')
265   print ('Valor minimo de lectura: '+str(minimo)+ 'V')
      crearTarget(max, min, periodo)
267
    def controlSistema2(temperatura):
269   ##ADC.read_raw(sensorPin2600)  ## primera medida erronea
      value = readADC()
271   if(value<1):
        value=1
273   ##print ('value:\t'+str(value))
      captura = datetime.now();
275
      ## Caclulo del valor de resistencia interna del sensor
277   ## el valor devuelto por read_raw ata en mV por eso divide entre 1000
      RStgs2600 = ((Vc*Rl)/(value/1000.0)) - Rl
279



281
      ## Se muestra por pantalla los datos
283   ##print ('Muestra PID['+str(0)+']\tTarget: '+str(value)+'\tValor:'+str(value)+'
        mV\tRs: '+str(RStgs2600)+'\tTemperatura: '+str(temperatura)+'\t'+str(captura)
        +'\n')

285   ## Se guardan datos en archivo
      ##f = open(rutaArchivo,"a")
287   ##datos = open(rutaArchivoDatos,"a")
      ##wline_datos = str(0)+'\t'+str(value)+'\t'+str(value)+'\t'+str(RStgs2600)+'\t
        '+str(temperatura)+'\t'+str(temperatura)+'\t'+str(captura)+'\n'
289   ##wline_f = 'Muestra PID['+str(0)+']\tTarget: '+str(value)+'\tValor:'+str(value
        )+'mV\tRs: '+str(RStgs2600)+'\tTemperatura: '+str(temperatura)+'\t'+str(
        captura)+'\n'
      ##datos.writelines(wline_datos)
291   ##datos.flush()
      ##f.writelines(wline_f)
293   ##f.flush()

295 def calentamientoPrevio(vectorT,vectorTd,valvula):
      contador = 0
297   ciclo = 0
      abrirElectrovalvula(valvula)
299   while ciclo < numCiclos:
        PWM.set_duty_cycle(heatPin2600,vectorTd[0])
301     while contador<20:
          tick = time.time()
303       ##print("Temperatura:\t"+str(vectorTd[0]))
          controlSistema2(vectorTd[0])
305       contador+=1
          tack = time.time();
307       time.sleep(SLEEP-(tack-tick))

309     contador=0
        while contador < len(vectorTd):
311       PWM.set_duty_cycle(heatPin2600,vectorTd[contador])
          tick = time.time()
313       ##print("Temperatura:\t"+str(vectorTd[contador]))
          controlSistema2(vectorTd[contador])
315       contador+=1
          tack = time.time();
```

```
317         time.sleep(SLEEP-(tack-tick))

319       contador=0
          PWM.set_duty_cycle(heatPin2600,vectorTd[19])
321       while contador<20:
            tick = time.time()
323         ##print("Temperatura:\t"+str(vectorTd[19]))
            controlSistema2(vectorTd[19])
325         contador+=1
            tack = time.time();
327         time.sleep(SLEEP-(tack-tick))

329       contador=0
          while contador < len(vectorT):
331         PWM.set_duty_cycle(heatPin2600,vectorT[contador])
            tick = time.time()
333         ##print("Temperatura:\t"+str(vectorT[contador]))
            controlSistema2(vectorT[contador])
335         contador+=1
            tack = time.time();
337         time.sleep(SLEEP-(tack-tick))

339       contador=0
          ciclo+=1
341
    def pidController(Kp, Ki, kd,SetPoint,SensorValue, lastError,lastTime, addError):
343   """
        deltaU =Kp*(ep + Kd*ed + (1/Ki)*ei)
345   """
      sampleTime = 1;
347
      error = round((SetPoint - SensorValue),3) ## cambio
349   currentTime = time.time()
      deltaTime = 1.0##currentTime - lastTime
351   deltaError = error - lastError


353
      ep = error
355   ed = (deltaError)/(deltaTime)
      ei = addError + (error*deltaTime)
357
      output = Kp *(ep +(Kd*ed) + ((1/Ki)*ei))
359
      datos = [output,error,ei,currentTime]
361   print(datos)
      return datos
363
    """Funcion que se encarga de llevar a acabo el experimento"""
365 def controlSistema(contador, subtarget,muestra,lastError,lastTime,addError,temp,
        flagTarget):
      ##ADC.read_raw(sensorPin2600)  ## primera medida erronea
367   global maximo
      global minimo
369   global temPIDmax
      global temPIDmin
371
      flagMaximo = False;  ## para cuando es el maximo
373   flagMinimo = False;  ## para cuando es el minimo


375
      value = readADC()
377   if(value<1):
        value=1
```

```
379   print ('value:\t'+str(value))
      captura = datetime.now();
381
      print ("LasError:"+str(lastError)+"\t lastTime:"+str(lastTime)+"\t addError:"+
        str(addError)+"\t Temperatura:"+str(temp)+"\t FlagTarget:"+str(flagTarget))
383
      ##value /=1000.0
385   ## actualizo los valores maximos y minimos de la senal
      if subtarget == max(target):
387     flagMaximo  = True;
        print ('Entro al maximo '+ str(flagMaximo))
389     e = subtarget*errorControl;
        print ('error:\t'+str(e))
391     if not(value>=(subtarget-e) and value<=(subtarget+e)):
          print ('El maximo cambia')
393       maximo = value
        else:
395       maximo = max(target)
      if subtarget == min(target):
397     flagMinimo= True
        print ('Entro al minimo ' +str(flagMinimo))
399     e = subtarget*errorControl;
        print('error:\t'+str(e))
401     if not(value>=(subtarget-e) and value<=(subtarget+e)):
          print ('El minimo cambia')
403       minimo = value
        else:
405       minimo = min(target)


407
      ## creo el parametro pid para llamar a la debida funcion
409   print ('sutarget:\t'+str(subtarget))

411   datosOut = pidController(Kp, Ki, Kd,subtarget,value, lastError,lastTime,
        addError)
      ##pid.SetPoint = subtarget
413
      ##pid.update(value)
415   errorMax = max(target)
      ##print("Error Maximo: "+str(errorMax))
417   errorMin =0;
      ##print("Error Minimo: "+str(errorMin))
419


421   """print(str(pid.Kp)+"\t"+str(pid.Ki)+"\t"+str(pid.Kd)+"\n"\
      +str(pid.sample_time)+"\t"+str(pid.current_time)+"\t"+str(pid.last_time)+"\n"\
423   +str(pid.last_error)+"\n"
      +str(pid.PTerm)+"\t"+str(pid.ITerm)+"\t"+str(pid.DTerm));"""
425
      Vout = datosOut[0]##pid.output/1000.0
427   error = datosOut[1]
      print("pid (mv):\t"+str(datosOut[0])+" error:\t"+str(error))
429
      if (error >=-0.5 and error <= 0.5):##cambio error ==0
431     t1 = 0##temp
      ##elif Vout>0:
433   else:
        t1 = (Vout-errorMax)*((temperaturaMin-temperaturaMax)/(errorMin-errorMax))+
        temperaturaMax
435     ##temperaturaPID = (Vout*100.0)/5.0
      """else:
437     Vp = ((Vout)*-1)+errorMax;
```

```
    t1 = abs(((Vp-errorMax)*((temperaturaMin-temperaturaMax)/(errorMin-errorMax))
    +temperaturaMax)-temp)"""
439
    t1 = round(t1,2)
441 print ('temperatura PID:\t'+str(t1))

443 """if((temp == temperaturaMax and Vout ==0)):
      temperaturaPID = temp-t1
445 elif (Vout >= 1 and Vout <= 2):
      temperaturaPID = temp-t1
447 else:
      temperaturaPID = temp+t1"""
449
    if(temp == temperaturaMax and error==0):
451   temperaturaPID = temp
    elif(temp == temperaturaMax and error>0):
453   temperaturaPID = temp
    elif(temp == temperaturaMax and error<0):
455   if(t1>0):
        temperaturaPID = temp - t1
457   else:
        temperaturaPID = temp + t1
459
    elif(temp < temperaturaMax and error==0):
461   temperaturaPID = temp
    elif(temp < temperaturaMax and error>0):
463   temperaturaPID = temp +t1
    elif(temp < temperaturaMax and error<0):
465   if(t1>0):
        temperaturaPID = temp - t1
467   else:
        temperaturaPID = temp + t1
469
    elif(temp > temperaturaMax and  (error==0 or error > 0)):
471   temperaturaPID = temp
    elif(temp > temperaturaMax and error<0):
473   temperaturaPID = temp + t1

475 """elif (Vout >= 1 and Vout <= 3):
      temperaturaPID = temp - (temp*(0.01))"""
477
    temperaturaPID = round(temperaturaPID,2)
479
    if temperaturaPID > temperaturaMax:
481   print ('temperatura PID mayor a max:\t'+str(temperaturaPID))
      temperaturaPID = temperaturaMax
483 elif temperaturaPID < temperaturaMin:
      print ('temperatura PID menor a min:\t'+str(temperaturaPID))
485   temperaturaPID = temperaturaMin

487 datosOut.append(temperaturaPID)
    ## actualizo la temperatura del sensor
489 PWM.set_duty_cycle(heatPin2600, temperaturaPID);


491
    muestrasTGS.append(value)
493 temperaturaTGS.append(temperaturaPID)

495 ## Caclulo del valor de resistencia interna del sensor
    ## el valor devuelto por read_raw ata en mV por eso divide entre 1000
497 RStgs2600 = ((Vc*Rl)/(value/1000.0)) - Rl
    RStgs2600 = round(RStgs2600,2)
499 ##rS_TGS.append(tgs2600RS)
```

```
501   ## Se muestra por pantalla los datos
      print ('Muestra PID['+str(contador)+']\tTarget: '+str(subtarget)+'\tValor:'+str
        (value)+'V\tRs: '+str(RStgs2600)+'\tTemperatura: '+str(temp)+'\tTemperatura: '
        +str(temperaturaPID)+'\t'+str(captura)+'\n')
503
      ## Se guardan datos en archivo
505   f = open(rutaArchivo,"a")
      datos = open(rutaArchivoDatos,"a")
507   wline_datos = str(contador)+'\t'+str(subtarget)+'\t'+str(value)+'\t'+str(
        RStgs2600)+'\t'+str(temp)+'\t'+str(temperaturaPID)+'\t'+str(captura)+'\n'
      wline_f = 'Muestra PID['+str(contador)+']\n>>Target: '+str(subtarget)+'V >>
        Valor:'+str(value)+'V >> Rs: '+str(RStgs2600)+' Temperatura: '+str(temp)+' '+'
         TemperaturaPID: '+str(temperaturaPID)+' '+str(captura)+'\n'
509   datos.writelines(wline_datos)
      datos.flush()
511   f.writelines(wline_f)
      f.flush()
513
      if(flagMaximo):
515     temPIDmax = temperaturaPID
        print("Temperatura del maximo: " +str(temPIDmax)+"\n")
517
      if (flagMinimo):
519     temPIDmin = temperaturaPID
        print("Temperatura del minimo: " +str(temPIDmin)+"\n")
521

523   ## se calcula el nuevo target cada vez que se termine el periodo
      if(flagTarget):
525
        print('Valores previos: maximo:\t'+str(maximo)+'\t minimo:\t'+str(minimo))
527     print('Valores previos: Tmax:\t'+str(temPIDmax)+'\t Tmin:\t'+str(temPIDmin))
        print('max(target)- maximo =\t'+str(max(target)- maximo))
529     print('maximo - minimo =\t'+str(maximo- minimo))
        ## guardo los valores temporales del maximo y minimo
531     maxAux = 0.0
        minAux = 0.0
533     ##primero compruebo que la amplitud de la senal sea superior a un valor
        if((maximo-minimo)<150.0):
535
          if(maximo <= minimo):
537         maxAux =  ceil(minimo)
            minAux = ceil(minimo - (minimo*0.7))
539       else:
            ## El maximo no llega a ser cercano al maximo del target
541         if((max(target)- maximo)>=0):
              if(temPIDmax > tempMaxMaxControl):
543             maxAux = ceil(maximo-(maximo*0.1))
              elif (temPIDmax < tempMaxMinControl):
545             maxAux = ceil (maximo +(maximo*0.1))
              else:
547             maxAux = ceil(maximo)
            else:
549           if(temPIDmax > tempMaxMaxControl):
                maxAux = ceil(maximo-(maximo*0.1))
551           elif (temPIDmax < tempMaxMinControl):
                maxAux = ceil (maximo +(maximo*0.1))
553           else:
                maxAux = ceil(maximo)
555
            ## Se define el valor del minimo
557         if((min(target)-minimo)<=0):
```

```
            if (temPIDmin > tempMinMaxControl):
559           minAux = ceil(maxAux -(maxAux*0.9))
            elif (temPIDmin < tempMinMinControl):
561           minAux = ceil(minimo + (maxAux*0.05))
            else:
563           minAux = ceil(maxAux -(maxAux*0.5))
          else:
565         if (temPIDmin > tempMinMaxControl):
              minAux = ceil(maxAux -(maxAux*0.9))
567         elif (temPIDmin < tempMinMinControl):
              minAux = ceil(minimo + (maxAux*0.05))
569         else:
              minAux = ceil(maxAux -(maxAux*0.5))
571
        ## COMPRUEBO QUE EL VALOR DEL MINIMO NUNCA SERA MENOR DE 100
573     if(minAux<100.0):
          minAux = 100.0
575       if(maxAux <= (minAux + minAux*0.25)):
            maxAux = maxAux +(maxAux*0.25)
577


579   else:
        if (temPIDmin > tempMinMaxControl):
581       minAux = ceil(min(target)-(min(target)*0.05)) ##minimo - (minimo*0.1)
        elif (temPIDmin < tempMinMinControl):
583       minAux = ceil(min(target)+(min(target)*0.05)) ##minimo + (minimo*0.1)
        else:
585       minAux = minimo


587
        if((max(target)- maximo)>0):
589       maxAux = ceil(maximo)
          ##if(temPIDmax < tempMaxMinControl):
591       ##   maximo = maximo + (maximo*0.05)
          ##else:
593       ##   maximo = maximo
        else:
595       if(temPIDmax > tempMaxMaxControl):
            maxAux = ceil(max(target) -max(target)*0.02)
597       elif (temPIDmax < tempMaxMinControl):
            maxAux = ceil(max(target) +max(target)*0.02)
599
        ##if(temPIDmin == temPIDmax):
601     ##   maximo = ceil(maximo +(maximo*0.3))
        ##   minimo = minimo -(minimo*0.3)
603
      ## compruebo que el maximo siempre sea mayor que el minimo
605   if(maxAux < minAux):
        aux = maxAux
607     maxAux = minAux
        minAux = maxAux
609
      if(maxAux >1800.0):
611     maxAux = 1800.0

613   if(minAux < 0.0):
        minAux = 1800.0*0.01
615



617
      crearTarget(maxAux,minAux,periodo)
619   flagMaximo = False
      flagMinimo = False
```

```
621
    return datosOut
623
def finExperimento():
625   PWM.stop(heatPin2600)
    PWM.stop(motorPin)
627   PWM.cleanup()
    ##_thread.exit()
629   cerrarElectrovalvula()

631   fecha_fin=datetime.now()
    print('\nEXPERIMENTO FINALIZADO CON EXITO')
633   print ('Experimento terminado: '+str(fecha_fin))
    f = open(rutaArchivo, "a")
635   f.write('\nEXPERIMENTO FINALIZADO CON EXITO\n')
    f.write('Fecha y hora de fin de experimentacion: ' +str(fecha_fin))
637   f.flush()
    f.close()
639   datos.close()
    ##tempH.close()
641
def main():
643   ## varaibles de control global
    global temPIDmax
645   global temPIDmin
    lastError =0;
647   addError =0;
    datos =[]
649   print ('*********************************************************\n'\
    '          ALGORITMO DE MODULACION EN LAZO CERRADO CONTROL PID \n'\
651   '*********************************************************\n' )

653   if(len(sys.argv)<8):
      print ('\n NUMERO DE PARAMETROS INCORRECTOS\n' \
655     'Los parametros que se deben pasar son:\n'\
      'Succion del motor (65% -100%)\n'
657     'Temperatura inicial TGS2600 (50%- 100%)\n'\
      'Tiempo de experimentacion (minutos)\n'\
659     'Tiempo de conmutacion  electrovalvulas (segundos)\n')
      return 0
661
    ## Succion del motor 50-100%
663   succion = float(sys.argv[1])
    if succion>100 or succion<65:
665     print ('PARAMETRO DE SUCCION DE MOTOR INCORRECTO.\n' \
      'Los parametros que se deben pasar son:\n'\
667     'Succion del motor (65% -100%)\n'
      'Temperatura inicial TGS2600 (50%- 100%)\n'\
669     'Tiempo de experimentacion (minutos)\n'\
      'Tiempo de conmutacion  electrovalvulas (segundos)\n')
671     return 0

673   ## Temperatura media del sensor TGS2600
    heat2600 = float(sys.argv[2])
675   if heat2600 >100 or heat2600 <0:
      print ('PARAMETRO DE CALENTAMIENTO DEL SENSOR  INCORRECTO.\n' \
677     'Los parametros que se deben pasar son:\n'\
      'Succion del motor (50% -100%)\n'
679     'Temperatura inicial TGS2600 (50%- 100%)\n'\
      'Tiempo de experimentacion (minutos)\n'\
681     'Tiempo de conmutacion  electrovalvulas (segundos)\n')
      return 0
683
```

```
## Duracion del experimento
tiempo = float(sys.argv[3])
if tiempo <1:
  print ('EL TIEMPO DE EXPERIMENTACION ES INCORECTO.\n' \
  'Los parametros que se deben pasar son:\n'\
  'Succion del motor (50% -100%)\n'
  'Temperatura inicial TGS2600 (50%- 100%)\n'\
  'Tiempo de experimentacion (minutos)\n'\
  'Tiempo de conmutacion  electrovalvulas (segundos)\n')
  return 0
## Tiempo conmutacion electrovalvula
cambio = float(sys.argv[4])
if cambio <1:
  print ('EL TIEMPO DE CONMUTACION DE VALVULAS ES INCORRECTO.\n' \
  'Los parametros que se deben pasar son:\n'\
  'Succion del motor (50% -100%)\n'
  'Temperatura inicial TGS2600 (50%- 100%)\n'\
  'Tiempo de experimentacion (minutos)\n'\
  'Tiempo de conmutacion  electrovalvulas (segundos)\n')
  return 0

odorante = float(sys.argv[5])
if odorante <1 and odorante>5:
  print ('LA ELECTROVALVULA SELECCIONADA NO ES CORRECTA.\n' \
  'Los parametros que se deben pasar son:\n'\
  'Succion del motor (50% -100%)\n'
  'Temperatura inicial TGS2600 (50%- 100%)\n'\
  'Tiempo de experimentacion (minutos)\n'\
  'Tiempo de conmutacion  electrovalvulas (segundos)\n'\
  'Electrovalvula incorrecta, valor entre 1 y 4\n')
  return 0

pico = float(sys.argv[6])
if pico<0:
  print ('VALOR MAXIMO INCORRECTO.\n' \
  'Los parametros que se deben pasar son:\n'\
  'Succion del motor (50% -100%)\n'
  'Temperatura inicial TGS2600 (50%- 100%)\n'\
  'Tiempo de experimentacion (minutos)\n'\
  'Tiempo de conmutacion  electrovalvulas (segundos)\n'\
  'Electrovalvula incorrecta, valor entre 1 y 4\n')
  return 0

bajo = float(sys.argv[7])
if bajo<0:
  print ('VALOR MAXIMO INCORRECTO.\n' \
  'Los parametros que se deben pasar son:\n'\
  'Succion del motor (50% -100%)\n'
  'Temperatura inicial TGS2600 (50%- 100%)\n'\
  'Tiempo de experimentacion (minutos)\n'\
  'Tiempo de conmutacion  electrovalvulas (segundos)\n'\
  'Electrovalvula incorrecta, valor entre 1 y 4\n')
  return 0



## Vector que contiene el orden de conmutacion de las electrovalvulas
vectorValvulas = [1, 1, 2, 2, 3, 3, 4, 4, 2, 1, 3, 1, 4, 3, 2, 4, 1]#
  #[1,2,3]##[int(odorante)]##
##numpy.random.randint(1,5,int((tiempo*60)/cambio)) ##[int(odorante)] ## numpy.
  random.randint(1,5,int((tiempo*60)/cambio))

##vectorValvulas =[4,4,4]
```

```
745
    vectorT= range(0,100,5)
747 vectorTd= range(95,-5,-5)

749 ## Se configura los puertos ADC, PWM
    motorStart(succion)
751 ADC.setup()
    PWM.start(heatPin2600,heat2600)
753


755 samples = 0
    while samples < 30:
757   tick = time.time()
      print ('Medidas de estabilizacion:\t'+str(readADC())) ## primera medida
      erronea
759   samples+=1
      tack = time.time();
761   time.sleep(SLEEP-(tack-tick))

763 ## precalentamiento del sensor
    calentamientoPrevio(vectorT,vectorTd,int(odorante))
765
    ## Generar Target
767 generarTarget(heat2600, periodo,pico,bajo)
    print(str(heat2600))
769 PWM.set_duty_cycle(heatPin2600, heat2600);
    esperar(3)
771
    ## Llamada al hilo para la medida de temperatura y humedad
773 ##_thread.start_new_thread(sensorTyH,(tiempo,))

775 ## creacion archivo de informacion experimento
    fileInfoTGS2600(vectorValvulas, succion, heat2600, tiempo, target,Kp,Ki,Kd,
      numCiclos,periodo)
777
    ## Sistema de control
779 contador = 1
    muestra = 0;
781 pos = 0 ;   ## cambiar este valor solo es una prueba "aire"
    temp = heat2600;
783 flagTarget = False
    temPIDmax =0.0
785 temPIDmin =0.0
    """print(str(pid.Kp)+"\t"+str(pid.Ki)+"\t"+str(pid.Kd)+"\n"\
787 +str(pid.sample_time)+"\t"+str(pid.current_time)+"\t"+str(pid.last_time)+"\n"\
    +str(pid.last_error)+"\n"
789 +str(pid.PTerm)+"\t"+str(pid.ITerm)+"\t"+str(pid.DTerm));
    tempActual = heat2600"""
791 lastTime = time.time()
    while contador < tiempo*60:
793
      electrovalvula = vectorValvulas[pos]
795   if electrovalvula == 1:
        print ('ELECTROVALVULA: '+str(electrovalvula)+' METANOL\n')
797   elif electrovalvula == 2:
        print ('ELECTROVALVULA: '+str(electrovalvula)+' ETANOL \n')
799   elif electrovalvula == 3:
        print ('ELECTROVALVULA: '+str(electrovalvula)+' BUTANOL \n')
801   else:
        print ('ELECTROVALVULA: '+str(electrovalvula)+' AIRE \n')
803   abrirElectrovalvula(electrovalvula)

805   j = 0;
```

```
    while j < cambio:
807   tick = time.time()
      if muestra == 0:
809     subtarget = target[muestra]
        muestra+=1
811     flagTarget = False
      elif (muestra%(periodo-1) ==0):
813     subtarget = target[periodo-1]
        muestra = 0
815     flagTarget = True
      else:
817     subtarget = target[muestra]
        muestra+=1
819     flagTarget = False


821


823
      datos = controlSistema(contador, subtarget,muestra,lastError,lastTime,
    addError,temp,flagTarget)
825   print ("temPIDmax:" +str(temPIDmax)+"\t temPIDmin:"+str(temPIDmin)+"\n")
      lastTime = datos[3]
827   addError = datos[2]
      lastError = datos[1]
829   temp = datos[4]
      contador+=1
831   j+=1
      tack = time.time();
833   time.sleep(SLEEP-(tack-tick))
    pos+=1
835
  finExperimento()
837 return 0

839 if __name__ == '__main__':
  try:
841   main()
  except KeyboardInterrupt:
843   print ('Interrupted')
      finExperimento()
845   sys.exit(0)
```

resources/principal_v05.5.py

### I.5.3. Código para la segunda versión de la placa de experimentación del controlador PID

```
## principal_v05.5
2 ## version que parte de principal_v05.4.py en la que se reliza
## Cambio del valor maximo de temeperatura que se aplicara al sensor
4 ## cambio en los valores  para cuando el aire esta en proceso
## de succion
6 import Adafruit_BBIO.ADC as ADC
import Adafruit_BBIO.GPIO as GPIO
8 import Adafruit_BBIO.PWM as PWM
import Adafruit_DHT
10 import sys
import os
12 import time
import math
14 import random
```

```
import numpy
from datetime import datetime, date
import _thread
from math import ceil

from  math import sin
from  math import pi

## Asignacion de puerto
sensorTemp22 = Adafruit_DHT.DHT22

Temp22      = 'P8_11' #Pin de lectura de temperatura/Humedad con DHT22 (Puerto GPIO_45)
electrovalve1 = 'P8_10' #Electrovalvula 1 (METANOL)
electrovalve2 = 'P8_12' #Electrovalvula 2 (ETANOL)
electrovalve3 = 'P8_14' #Electrovalvula 3 (BUTANOL)
electrovalve4 = 'P8_16' #Electrovalvula 4 (AIRE)
electrovalve5 = 'P8_18' #Electrovalvula 5 (METANOL)

motorPin    = 'P9_21' #Motor de succion
heatPin2600   = 'P9_22' #Calentamiento sensor TGS2600 (Puerto GPIO_2)
sensorPin2600 = 'P9_40' #Sensor TGS2600 (Puerto AIN_1)

## DECLARACION DE SALIDAS DEL SISTEMA.
GPIO.setup("P8_10",GPIO.OUT)
GPIO.setup("P8_12",GPIO.OUT)
GPIO.setup("P8_14",GPIO.OUT)
GPIO.setup("P8_16",GPIO.OUT)
GPIO.setup("P8_18",GPIO.OUT)

## DECLARACION VARIABLES DEL SISTEMA

SLEEP = 1     ## Periodo de estabilizacion  para cada adquisicion
target = []     ## Almaceno los valores de la senal objetivo
periodo = 50      ## Periodo de la senal Target (segundos)
errorControl = 0.05   ## Parametro para controlar el tracking en %0.3
temperaturaMax = 95 ## Ciclo pwm maximo temperatura sensor
temperaturaMin = 5 ## Cilco pwm minimo temperatura sensor

tempMaxMaxControl = 95 ## Ciclo pwm maximo temperatura sensor para controlar la senal target
tempMaxMinControl = 85 ## Ciclo pwm maximo temperatura sensor para controlar la senal target

tempMinMaxControl = 15 ## Cilco pwm minimo temperatura sensor para controlar la senal target
tempMinMinControl = 5 ## Cilco pwm minimo temperatura sensor para controlar la senal target

muestrasTGS =[]   ## Almacena las muestras de odorantes
temperaturaTGS =[]  ## Almacena la temperatura de calentamiento  del sensor
rS_TGS = []     ## Almacena la resistencia del sensor
maximo = 0      ## Almaceno el valor maximo de la senal objetivo
minimo =0     ## Alamceno el valor minimo de la senal objetivo
numCiclos = 1   ## Variable para controlar el numero de ciclos calentamiento previo del sistema

tempActual =0
lectura = 10;
alfa = 0.3

## VARIABLES SENSOR
Rl = 470  #omh
```

```
Vc = 5    #Voltios

SLEEP_tyh = 59  ##Medida de temperatura y humedad

## VARIABLES PARA EL PID
Kp = 0.6*alfa      ## Constante proporcional
Kd =  0.125*periodo##40.0/8          ## Constante derivativa
Ki = (0.5)*periodo##40.0/2           ## Constante integral


## Datos para el fichero de salida
nombreArchivo = "pidTgs2600.txt"
archivoDatos = "pidTgs2600.dat"
archivoTyH = "pidTgs2600TyH.data"
fileString = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime()) +'_'+
    nombreArchivo
fileStringDatos = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime()) +'_'+
    archivoDatos
fileStringTyH = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime()) +'_'+
    archivoTyH
actual = datetime.now()
ruta = "/media/sdcard/PIDCONTROL/"+str(actual.month)+"/"+str(actual.day)+"/"
if not os.path.exists(ruta): os.makedirs(ruta)

rutaArchivo = ruta.strip()+str(fileString)
rutaArchivoDatos = ruta.strip()+str(fileStringDatos)
rutaArchivoTyH = ruta.strip()+str(fileStringTyH)

## Se crean los ficheros de datos y temperatura
f = open(rutaArchivo,"w+")
datos = open(rutaArchivoDatos,"w+")
##tempH = open(rutaArchivoTyH,"w+")

def fileInfoTGS2600(vectorValvulas, succion, heat2600, tiempo, target,Kp,Ki,Kd,
    numCiclos,periodo):
  f = open(rutaArchivo,"w+")
  f.write ('/*************************************************************/')
  f.write ('Plataforma de experimentacion BBB\n')
  f.write ('Carlos Velasco\n')
  f.write ('Sensor TGS2600\n')
  f.write ('-modluacion en lazo cerrado PID\n')
  fecha = time.strftime("%a%d%b%Y-%HH%MM%SS", time.localtime())
  f.write ('Fecha y hora de inicio: ' + str(fecha) + '\n')
  f.write ('Ruta del fichero: ' + str(ruta) + '\n')
  f.write ('Nombre archivo: ' + str(fileString) + '\n')
  f.write ('Nombre archivo de datos: ' + str(fileStringDatos) + '\n')
  f.write ('Nombre archivo de TyH: ' + str(fileStringTyH) + '\n')
  f.write ('Conmutacion entre electrovalvulas: ' +str(vectorValvulas)+ '\n\n')
  f.write ('Parametros requeridos para el experimento \n')
  f.write ('Succion del motor (50-100) >>> '+ str(succion)+ '%\n')
  f.write ('Temperatura Promedio TGS2600 (50-100) >>> '+str(heat2600)+'% Pin PWM
    : ' +str(heatPin2600)+ '\n\n')
  f.write ('Control Proporcional: '+str(Kp)+'\n')
  f.write ('Control Integral: '+str(Ki)+'\n')
  f.write ('Control Derivativo: '+str(Kd)+'\n')
  f.write ('Ciclos de precalentamiento: '+str(numCiclos)+'\n')
  f.write ('Periodo: '+str(periodo)+'\n')
  f.write ('Duracion del experimento: ' +str(tiempo)+ ' minutos\n')
  f.write ('/*****************************************************\n')
  f.write ('Target Inicial:\n')
  f.write (str(target)+'\n')
  f.write ('DATOS OBTENIDOS:\n\n')
```

```
def sensorTyH(tiempo):
  j=0
  tiempo = round(tiempo)
  time.sleep(9)
  # proceso de lectura
  while(j<tiempo):
    tickHT = time.time()
    humedad, temperatura = Adafruit_DHT.read_retry(sensorTemp22, Temp22)
    ahora = datetime.now()
    tackHT = time.time()
    ticktack = tackHT-tickHT

    if ticktack >SLEEP_tyh:
      print ("Tiempo medicion H y T > SLEEP:", ticktack)
    else:
      print ("Tiempo medicion H y T:",ticktack)
      if humedad is not None and temperatura is not None:
        print ('\nSensor DHT22: ' +str(sensorTemp22))
        print ('>>> '+str(ahora)+' Temp = ' +str(temperatura)+ ' Humidity = ' +
    str(humedad)+'\n')
        tempH = open(rutaArchivoTyH, "a")
        wline_h = str(ahora)+'\t'+str(temperatura)+'\t'+str(humedad)+'\n'
        tempH.writelines(wline_h)
        tempH.flush()
      else:
        print ('Failed to get reading, Try again!')
        tempH = open(rutaArchivoTyH, "a")
        wline_h=str(ahora)+' '+str(temperatura)+' '+str(humedad)+'\n'
        tempH.writelines(wline_h)
        tempH.flush()

    tack=time.time()
    p = tack - tickHT;
    time.sleep(SLEEP_tyh-p)
    j+=1

"""**********************************************************************
                        FUNCIONES DEL SISTEMA
**********************************************************************"""

def abrirElectrovalvula(electrovalvula):
  if electrovalvula ==1:
    print ('electrovalvula 1\n')
    GPIO.output(electrovalve1, GPIO.LOW)
    GPIO.output(electrovalve2, GPIO.HIGH)
    GPIO.output(electrovalve3, GPIO.HIGH)
    GPIO.output(electrovalve4, GPIO.HIGH)
    GPIO.output(electrovalve5, GPIO.HIGH)
  elif electrovalvula == 2:
    print ('electrovalvula 2\n')
    GPIO.output(electrovalve1, GPIO.HIGH)
    GPIO.output(electrovalve2, GPIO.LOW)
    GPIO.output(electrovalve3, GPIO.HIGH)
    GPIO.output(electrovalve4, GPIO.HIGH)
    GPIO.output(electrovalve5, GPIO.HIGH)
  elif electrovalvula == 3:
    print ('electrovalvula 3\n')
    GPIO.output(electrovalve1, GPIO.HIGH)
    GPIO.output(electrovalve2, GPIO.HIGH)
    GPIO.output(electrovalve3, GPIO.LOW)
    GPIO.output(electrovalve4, GPIO.HIGH)
    GPIO.output(electrovalve5, GPIO.HIGH)
  elif electrovalvula == 4:
```

```
192     print ('electrovalvula 4\n')
        GPIO.output(electrovalve1, GPIO.HIGH)
194     GPIO.output(electrovalve2, GPIO.HIGH)
        GPIO.output(electrovalve3, GPIO.HIGH)
196     GPIO.output(electrovalve4, GPIO.LOW)
        GPIO.output(electrovalve5, GPIO.HIGH)
198   elif electrovalvula == 5:
        print ('electrovalvula 5\n')
200     GPIO.output(electrovalve1, GPIO.HIGH)
        GPIO.output(electrovalve2, GPIO.HIGH)
202     GPIO.output(electrovalve3, GPIO.HIGH)
        GPIO.output(electrovalve4, GPIO.HIGH)
204     GPIO.output(electrovalve5, GPIO.LOW)

206 def cerrarElectrovalvula():
      GPIO.output(electrovalve1, GPIO.LOW)
208   GPIO.output(electrovalve2, GPIO.LOW)
      GPIO.output(electrovalve3, GPIO.LOW)
210   GPIO.output(electrovalve4, GPIO.LOW)
      GPIO.output(electrovalve5, GPIO.LOW)
212
    def motorStart(succion):
214   PWM.start(motorPin,succion)

216 def readADC():
      valor = (1800*(ADC.read(sensorPin2600)))
218   valor = round(valor,3)
      return valor
220
    def crearTarget(Vmax, Vmin, periodo):
222     global target
        print('maximo:\t'+str(Vmax)+'\t minimo:\t'+str(Vmin))
224     """ Asen(BX+C)"""
        A = (Vmax - Vmin)/2.0
226     B = (2*pi)/(periodo) ## periodo

228     senal =[]

230     for x in range(periodo+1): ## periodo+1
          senal.append(round(((A*sin(B*x))+(A+Vmin)),3))
232
        target = senal[0:periodo] ## senal[1:periodo]
234     print (target)

236 def esperar(tiempo):
      time.sleep(tiempo)
238
    def mediaCalententamiento():
240   print ('Entro mediaCalententamiento')
      value = 0;
242   NM = 0;
      while NM <lectura:
244     tick = time.time()
        adcValue = readADC()
246     print ('adcValue: \t'+str(adcValue))
        value += adcValue
248     NM+=1
        tack = time.time();
250     time.sleep(SLEEP-(tack-tick))
      value/=lectura
252   value = round(value)
      print ('Media:\t'+str(value))
254   return value
```

```
256 def generarTarget(temperatura,periodo,max,min):
    global maximo
258 global minimo
    ##ADC.read_raw(sensorPin2600) ## primera medida erronea
260 abrirElectrovalvula(4) ## se abre electrovalvula del aire
    ## Se toman muestras para el valor maximo 80% (4V)
262 PWM.set_duty_cycle(heatPin2600, temperaturaMax);
    esperar(1)
264 ##ADC.read_raw(sensorPin2600) ## primera medida erronea
    ##maximo = ADC.read_raw(sensorPin2600)
266 ##maximo = mediaCalententamiento()

268 ## Se toman muestras para el valor minimo 10% (1V)
    PWM.set_duty_cycle(heatPin2600, temperaturaMin);
270 esperar(1)
    ##ADC.read_raw(sensorPin2600) ## primera medida erronea
272 ##minimo = ADC.read_raw(sensorPin2600)
    ##minimo = mediaCalententamiento()
274
    ## Escritua por pantalla de valores obtenidos
276 print ('Valor maximo de lectura: '+str(maximo)+ 'V')
    print ('Valor minimo de lectura: '+str(minimo)+ 'V')
278 crearTarget(max, min, periodo)

280 def controlSistema2(temperatura):
    ##ADC.read_raw(sensorPin2600) ## primera medida erronea
282 value = readADC()
    if(value<1):
284   value=1
    ##print ('value:\t'+str(value))
286 captura = datetime.now();

288 ## Caclulo del valor de resistencia interna del sensor
    ## el valor devuelto por read_raw ata en mV por eso divide entre 1000
290 RStgs2600 = ((Vc*Rl)/(value/1000.0)) - Rl


292



294 ## Se muestra por pantalla los datos
    ##print ('Muestra PID['+str(0)+']\tTarget: '+str(value)+'\tValor:'+str(value)+'
      mV\tRs: '+str(RStgs2600)+'\tTemperatura: '+str(temperatura)+'\t'+str(captura)
      +'\n')
296
    ## Se guardan datos en archivo
298 ##f = open(rutaArchivo,"a")
    ##datos = open(rutaArchivoDatos,"a")
300 ##wline_datos = str(0)+'\t'+str(value)+'\t'+str(value)+'\t'+str(RStgs2600)+'\t
      '+str(temperatura)+'\t'+str(temperatura)+'\t'+str(captura)+'\n'
    ##wline_f = 'Muestra PID['+str(0)+']\tTarget: '+str(value)+'\tValor:'+str(value
      )+'mV\tRs: '+str(RStgs2600)+'\tTemperatura: '+str(temperatura)+'\t'+str(
      captura)+'\n'
302 ##datos.writelines(wline_datos)
    ##datos.flush()
304 ##f.writelines(wline_f)
    ##f.flush()
306
    def calentamientoPrevio(vectorT,vectorTd,valvula):
308 contador = 0
    ciclo = 0
310 abrirElectrovalvula(valvula)
    while ciclo < numCiclos:
312   PWM.set_duty_cycle(heatPin2600,vectorTd[0])
```

```
    while contador<20:
314       tick = time.time()
          ##print("Temperatura:\t"+str(vectorTd[0]))
316       controlSistema2(vectorTd[0])
          contador+=1
318       tack = time.time();
          time.sleep(SLEEP-(tack-tick))
320
    contador=0
322    while contador < len(vectorTd):
         PWM.set_duty_cycle(heatPin2600,vectorTd[contador])
324       tick = time.time()
          ##print("Temperatura:\t"+str(vectorTd[contador]))
326       controlSistema2(vectorTd[contador])
          contador+=1
328       tack = time.time();
          time.sleep(SLEEP-(tack-tick))
330
    contador=0
332    PWM.set_duty_cycle(heatPin2600,vectorTd[19])
    while contador<20:
334       tick = time.time()
          ##print("Temperatura:\t"+str(vectorTd[19]))
336       controlSistema2(vectorTd[19])
          contador+=1
338       tack = time.time();
          time.sleep(SLEEP-(tack-tick))
340
    contador=0
342    while contador < len(vectorT):
         PWM.set_duty_cycle(heatPin2600,vectorT[contador])
344       tick = time.time()
          ##print("Temperatura:\t"+str(vectorT[contador]))
346       controlSistema2(vectorT[contador])
          contador+=1
348       tack = time.time();
          time.sleep(SLEEP-(tack-tick))
350
    contador=0
352    ciclo+=1

354 def pidController(Kp, Ki, kd,SetPoint,SensorValue, lastError,lastTime, addError):
  """
356    deltaU =Kp*(ep + Kd*ed + (1/Ki)*ei)
  """
358  sampleTime = 1;

360  error = round((SetPoint - SensorValue),3) ## cambio
  currentTime = time.time()
362  deltaTime = 1.0##currentTime - lastTime
  deltaError = error - lastError
364


366  ep = error
  ed = (deltaError)/(deltaTime)
368  ei = addError + (error*deltaTime)

370  output = Kp *(ep +(Kd*ed) + ((1/Ki)*ei))

372  datos = [output,error,ei,currentTime]
  print(datos)
374  return datos
```

```
376 """Funcion que se encarga de llevar a acabo el experimento"""
    def controlSistema(contador, subtarget,muestra,lastError,lastTime,addError,temp,
        flagTarget):
378   ##ADC.read_raw(sensorPin2600)  ## primera medida erronea
      global maximo
380   global minimo
      global temPIDmax
382   global temPIDmin

384   flagMaximo = False;  ## para cuando es el maximo
      flagMinimo = False;  ## para cuando es el minimo
386

388   value = readADC()
      if(value<1):
390     value=1
      print ('value:\t'+str(value))
392   captura = datetime.now();

394   print ("LasError:"+str(lastError)+"\t lastTime:"+str(lastTime)+"\t addError:"+
        str(addError)+"\t Temperatura:"+str(temp)+"\t FlagTarget:"+str(flagTarget))

396   ##value /=1000.0
      ## actualizo los valores maximos y minimos de la senal
398   if subtarget == max(target):
        flagMaximo  = True;
400     print ('Entro al maximo '+ str(flagMaximo))
        e = subtarget*errorControl;
402     print ('error:\t'+str(e))
        if not(value>=(subtarget-e) and value<=(subtarget+e)):
404       print ('El maximo cambia')
          maximo = value
406     else:
          maximo = max(target)
408   if subtarget == min(target):
        flagMinimo= True
410     print ('Entro al minimo ' +str(flagMinimo))
        e = subtarget*errorControl;
412     print('error:\t'+str(e))
        if not(value>=(subtarget-e) and value<=(subtarget+e)):
414       print ('El minimo cambia')
          minimo = value
416     else:
          minimo = min(target)
418


420   ## creo el parametro pid para llamar a la debida funcion
      print ('sutarget:\t'+str(subtarget))
422
      datosOut = pidController(Kp, Ki, Kd,subtarget,value, lastError,lastTime,
        addError)
424   ##pid.SetPoint = subtarget

426   ##pid.update(value)
      errorMax = max(target)
428   ##print("Error Maximo: "+str(errorMax))
      errorMin =0;
430   ##print("Error Minimo: "+str(errorMin))


432
      """print(str(pid.Kp)+"\t"+str(pid.Ki)+"\t"+str(pid.Kd)+"\n"\
434   +str(pid.sample_time)+"\t"+str(pid.current_time)+"\t"+str(pid.last_time)+"\n"\
      +str(pid.last_error)+"\n"
```

```
436   +str(pid.PTerm)+"\t"+str(pid.ITerm)+"\t"+str(pid.DTerm));"""

438   Vout = datosOut[0]##pid.output/1000.0
      error = datosOut[1]
440   print("pid (mv):\t"+str(datosOut[0])+" error:\t"+str(error))

442   if (error >=-0.5 and error <= 0.5):##cambio error ==0
        t1 = 0##temp
444   ##elif Vout>0:
      else:
446     t1 = (Vout-errorMax)*((temperaturaMin-temperaturaMax)/(errorMin-errorMax))+
        temperaturaMax
        ##temperaturaPID = (Vout*100.0)/5.0
448   """else:
        Vp = ((Vout)*-1)+errorMax;
450     t1 = abs(((Vp-errorMax)*((temperaturaMin-temperaturaMax)/(errorMin-errorMax))
        +temperaturaMax)-temp)"""

452   t1 = round(t1,2)
      print ('temperatura PID:\t'+str(t1))
454
      """if((temp == temperaturaMax and Vout ==0)):
456     temperaturaPID = temp-t1
      elif (Vout >= 1 and Vout <= 2):
458     temperaturaPID = temp-t1
      else:
460     temperaturaPID = temp+t1"""

462   if(temp == temperaturaMax and error==0):
        temperaturaPID = temp
464   elif(temp == temperaturaMax and error>0):
        temperaturaPID = temp
466   elif(temp == temperaturaMax and error<0):
        if(t1>0):
468       temperaturaPID = temp - t1
        else:
470       temperaturaPID = temp + t1

472   elif(temp < temperaturaMax and error==0):
        temperaturaPID = temp
474   elif(temp < temperaturaMax and error>0):
        temperaturaPID = temp +t1
476   elif(temp < temperaturaMax and error<0):
        if(t1>0):
478       temperaturaPID = temp - t1
        else:
480       temperaturaPID = temp + t1

482   elif(temp > temperaturaMax and  (error==0 or error > 0)):
        temperaturaPID = temp
484   elif(temp > temperaturaMax and error<0):
        temperaturaPID = temp + t1
486
      """elif (Vout >= 1 and Vout <= 3):
488     temperaturaPID = temp - (temp*(0.01))"""

490   temperaturaPID = round(temperaturaPID,2)

492   if temperaturaPID > temperaturaMax:
        print ('temperatura PID mayor a max:\t'+str(temperaturaPID))
494     temperaturaPID = temperaturaMax
      elif temperaturaPID < temperaturaMin:
496     print ('temperatura PID menor a min:\t'+str(temperaturaPID))
```

```
    temperaturaPID = temperaturaMin
498
    datosOut.append(temperaturaPID)
500 ## actualizo la temperatura del sensor
    PWM.set_duty_cycle(heatPin2600, temperaturaPID);
502

504 muestrasTGS.append(value)
    temperaturaTGS.append(temperaturaPID)
506
    ## Caclulo del valor de resistencia interna del sensor
508 ## el valor devuelto por read_raw ata en mV por eso divide entre 1000
    RStgs2600 = ((Vc*Rl)/(value/1000.0)) - Rl
510 RStgs2600 = round(RStgs2600,2)
    ##rS_TGS.append(tgs2600RS)
512
    ## Se muestra por pantalla los datos
514 print ('Muestra PID['+str(contador)+']\tTarget: '+str(subtarget)+'\tValor:'+str
      (value)+'V\tRs: '+str(RStgs2600)+'\tTemperatura: '+str(temp)+'\tTemperatura: '
      +str(temperaturaPID)+'\t'+str(captura)+'\n')

516 ## Se guardan datos en archivo
    f = open(rutaArchivo,"a")
518 datos = open(rutaArchivoDatos,"a")
    wline_datos = str(contador)+'\t'+str(subtarget)+'\t'+str(value)+'\t'+str(
      RStgs2600)+'\t'+str(temp)+'\t'+str(temperaturaPID)+'\t'+str(captura)+'\n'
520 wline_f = 'Muestra PID['+str(contador)+']\n>>Target: '+str(subtarget)+'V >>
      Valor:'+str(value)+'V >> Rs: '+str(RStgs2600)+' Temperatura: '+str(temp)+' '+'
       TemperaturaPID: '+str(temperaturaPID)+' '+str(captura)+'\n'
    datos.writelines(wline_datos)
522 datos.flush()
    f.writelines(wline_f)
524 f.flush()

526 if(flagMaximo):
      temPIDmax = temperaturaPID
528   print("Temperatura del maximo: " +str(temPIDmax)+"\n")

530 if (flagMinimo):
      temPIDmin = temperaturaPID
532   print("Temperatura del minimo: " +str(temPIDmin)+"\n")


534
    ## se calcula el nuevo target cada vez que se termine el periodo
536 if(flagTarget):

538   print('Valores previos: maximo:\t'+str(maximo)+'\t minimo:\t'+str(minimo))
      print('Valores previos: Tmax:\t'+str(temPIDmax)+'\t Tmin:\t'+str(temPIDmin))
540   print('max(target)- maximo =\t'+str(max(target)- maximo))
      print('maximo - minimo =\t'+str(maximo- minimo))
542   ## guardo los valores temporales del maximo y minimo
      maxAux = 0.0
544   minAux = 0.0
      ##primero compruebo que la amplitud de la senal sea superior a un valor
546   if((maximo-minimo)<100.0):
        print('maximo - minimo < 100\t')
548     if(maximo <= minimo):
          print('maximo <= minimo\t')
550       maxAux =  ceil(minimo)
          minAux = ceil(minimo - (minimo*0.7))
552       print(' ceil(minimo - (minimo*0.7)) =\t'+str(minAux))
        else:
554       ## El maximo no llega a ser cercano al maximo del target
```

```
        if((max(target)- maximo)>=0):
556       print('(max(target)- maximo)>=0\t')
          if(temPIDmax > tempMaxMaxControl):
558         print('temPIDmax > tempMaxMaxControl\t')
            maxAux = ceil(maximo-(maximo*0.1))
560         print('ceil(maximo-(maximo*0.1))=\t'+str(maxAux))
          elif (temPIDmax < tempMaxMinControl):
562         print('temPIDmax < tempMaxMinControl\t')
            maxAux = ceil (maximo +(maximo*0.1))
564         print('ceil (maximo +(maximo*0.1)=\t'+str(maxAux))
          else:
566         print('else:\t')
            maxAux = ceil(maximo)
568         print('ceil(maximo)=\t'+str(maxAux))
        else:
570       print('else (max(target)- maximo)>=0\t')
          if(temPIDmax > tempMaxMaxControl):
572         print('temPIDmax > tempMaxMaxControl\t')
            maxAux = ceil(maximo-(maximo*0.1))
574         print('ceil(maximo-(maximo*0.1))=\t'+str(maxAux))
          elif (temPIDmax < tempMaxMinControl):
576         print('temPIDmax < tempMaxMinControl\t')
            maxAux = ceil (maximo +(maximo*0.1))
578         print('ceil (maximo +(maximo*0.1)))=\t'+str(maxAux))
          else:
580         print('else:\t')
            maxAux = ceil(maximo)
582         print('ceil(maximo)=\t'+str(maxAux))

584     ## Se define el valor del minimo
        if((min(target)-minimo)<=0):
586       print('(min(target)-minimo)<=0\t')
          if (temPIDmin > tempMinMaxControl):
588         print('temPIDmin > tempMinMaxControl\t')
            minAux = ceil(maxAux -(maxAux*0.9))
590         print('ceil(maxAux -(maxAux*0.9))=\t'+str(minAux))
          elif (temPIDmin < tempMinMinControl):
592         print('temPIDmin < tempMinMinControl\t')
            minAux = ceil(minimo + (maxAux*0.05))
594         print('ceil(minimo + (maxAux*0.05))=\t'+str(minAux))
          else:
596         print('else:\t')
            minAux = ceil(maxAux -(maxAux*0.5))
598         print('ceil(maxAux -(maxAux*0.5))\t'+str(minAux))
        else:
600       print('else (min(target)-minimo)<=0\t')
          if (temPIDmin > tempMinMaxControl):
602         print('temPIDmin > tempMinMaxControl\t')
            minAux = ceil(maxAux -(maxAux*0.9))
604         print('ceil(maxAux -(maxAux*0.9))=\t'+str(minAux))
          elif (temPIDmin < tempMinMinControl):
606         print('temPIDmin < tempMinMinControl\t')
            minAux = ceil(minimo + (maxAux*0.05))
608         print('ceil(minimo + (maxAux*0.05))=\t'+str(minAux))
          else:
610         print('else:\t')
            minAux = ceil(maxAux -(maxAux*0.5))
612         print('ceil(maxAux -(maxAux*0.5))=\t'+str(minAux))

614     ## COMPRUEBO QUE EL VALOR DEL MINIMO NUNCA SERA MENOR DE 100
        if(minAux<100.0):
616       print('minAux<100.0')
          minAux = 100.0
```

```
618           if(maxAux <= (minAux + minAux*0.25)):
                print('maxAux <= (minAux + minAux*0.25)\t')
620             maxAux = maxAux +(maxAux*0.25)
                print(' maxAux +(maxAux*0.25)=\t'+str(maxAux))
622
624       else:
            print('else maximo - minimo < 100\t')
626         if (temPIDmin > tempMinMaxControl):
              print('temPIDmin > tempMinMaxControl\t')
628           minAux = ceil(min(target)-(min(target)*0.05)) ##minimo - (minimo*0.1)
              print('ceil(min(target)-(min(target)*0.05))=\t'+str(minAux))
630         elif (temPIDmin < tempMinMinControl):
              print('temPIDmin < tempMinMinControl\t')
632           minAux = ceil(min(target)+(min(target)*0.05)) ##minimo + (minimo*0.1)
              print('ceil(min(target)+(min(target)*0.05)) =\t'+str(minAux))
634         else:
              print('else\t')
636           minAux = minimo
              print('minAux = minimo \t'+str(minAux))
638
            if((max(target)- maximo)>0):
640           print('(max(target)- maximo)>0\t')
              maxAux = ceil(maximo)
642           print('maxAux = ceil(maximo)\t'+str(maxAux))
              ##if(temPIDmax < tempMaxMinControl):
644           ##  maximo = maximo + (maximo*0.05)
              ##else:
646           ##  maximo = maximo
            else:
648           print('else (max(target)- maximo)>0\t')
              if(temPIDmax > tempMaxMaxControl):
650             print('temPIDmax > tempMaxMaxControl\t')
                maxAux = ceil(max(target) -max(target)*0.02)
652             print('ceil(max(target) -max(target)*0.02)=\t'+str(maxAux))
              elif (temPIDmax < tempMaxMinControl):
654             print('temPIDmax < tempMaxMinControlt')
                maxAux = ceil(max(target) +max(target)*0.02)
656             print('ceil(max(target) +max(target)*0.02)=\t'+str(maxAux))
              else:
658             print('else\t')
                maxAux = ceil(maximo)
660             print('maxAux = ceil(maximo)=\t'+str(maxAux))
          ##if(temPIDmin == temPIDmax):
662       ##  maximo = ceil(maximo +(maximo*0.3))
          ##  minimo = minimo -(minimo*0.3)
664
        ## compruebo que el maximo siempre sea mayor que el minimo
666     if(maxAux < minAux):
          print('maxAux < minAux\t')
668       aux = maxAux
          maxAux = minAux
670       minAux = maxAux
          print('maxAux\=t'+str(maxAux)+'\tminAux=\t'+str(minAux))
672
        if(maxAux >1800.0):
674       maxAux = 1800.0

676     if(minAux < 0.0):
          minAux = 1800.0*0.01
678
680
```

```
    crearTarget(maxAux,minAux,periodo)
682     flagMaximo = False
        flagMinimo = False
684
    return datosOut
686
def finExperimento():
688   PWM.stop(heatPin2600)
      PWM.stop(motorPin)
690   PWM.cleanup()
      ##_thread.exit()
692   cerrarElectrovalvula()

694   fecha_fin=datetime.now()
      print('\nEXPERIMENTO FINALIZADO CON EXITO')
696   print ('Experimento terminado: '+str(fecha_fin))
      f = open(rutaArchivo, "a")
698   f.write('\nEXPERIMENTO FINALIZADO CON EXITO\n')
      f.write('Fecha y hora de fin de experimentacion: ' +str(fecha_fin))
700   f.flush()
      f.close()
702   datos.close()
      ##tempH.close()
704
def main():
706   ## varaibles de control global
      global temPIDmax
708   global temPIDmin
      lastError =0;
710   addError =0;
      datos =[]
712   print ('*********************************************************\n'\
      '          ALGORITMO DE MODULACION EN LAZO CERRADO CONTROL PID \n'\
714   '*********************************************************\n' )

716   if(len(sys.argv)<8):
        print ('\n NUMERO DE PARAMETROS INCORRECTOS\n' \
718     'Los parametros que se deben pasar son:\n'\
        'Succion del motor (65% -100%)\n'
720     'Temperatura inicial TGS2600 (50%- 100%)\n'\
        'Tiempo de experimentacion (minutos)\n'\
722     'Tiempo de conmutacion  electrovalvulas (segundos)\n')
        return 0
724
      ## Succion del motor 50-100%
726   succion = float(sys.argv[1])
      if succion>100 or succion<65:
728     print ('PARAMETRO DE SUCCION DE MOTOR INCORRECTO.\n' \
        'Los parametros que se deben pasar son:\n'\
730     'Succion del motor (65% -100%)\n'
        'Temperatura inicial TGS2600 (50%- 100%)\n'\
732     'Tiempo de experimentacion (minutos)\n'\
        'Tiempo de conmutacion  electrovalvulas (segundos)\n')
734     return 0

736   ## Temperatura media del sensor TGS2600
      heat2600 = float(sys.argv[2])
738   if heat2600 >100 or heat2600 <0:
        print ('PARAMETRO DE CALENTAMIENTO DEL SENSOR  INCORRECTO.\n' \
740     'Los parametros que se deben pasar son:\n'\
        'Succion del motor (50% -100%)\n'
742     'Temperatura inicial TGS2600 (50%- 100%)\n'\
        'Tiempo de experimentacion (minutos)\n'\
```

```
744     'Tiempo de conmutacion  electrovalvulas (segundos)\n')
        return 0
746
    ## Duracion del experimento
748 tiempo = float(sys.argv[3])
    if tiempo <1:
750   print ('EL TIEMPO DE EXPERIMENTACION ES INCORECTO.\n' \
      'Los parametros que se deben pasar son:\n'\
752   'Succion del motor (50% -100%)\n'
      'Temperatura inicial TGS2600 (50%- 100%)\n'\
754   'Tiempo de experimentacion (minutos)\n'\
      'Tiempo de conmutacion  electrovalvulas (segundos)\n')
756   return 0
    ## Tiempo conmutacion electrovalvula
758 cambio = float(sys.argv[4])
    if cambio <1:
760   print ('EL TIEMPO DE CONMUTACION DE VALVULAS ES INCORRECTO.\n' \
      'Los parametros que se deben pasar son:\n'\
762   'Succion del motor (50% -100%)\n'
      'Temperatura inicial TGS2600 (50%- 100%)\n'\
764   'Tiempo de experimentacion (minutos)\n'\
      'Tiempo de conmutacion  electrovalvulas (segundos)\n')
766   return 0

768 odorante = float(sys.argv[5])
    if odorante <1 and odorante>5:
770   print ('LA ELECTROVALVULA SELECCIONADA NO ES CORRECTA.\n' \
      'Los parametros que se deben pasar son:\n'\
772   'Succion del motor (50% -100%)\n'
      'Temperatura inicial TGS2600 (50%- 100%)\n'\
774   'Tiempo de experimentacion (minutos)\n'\
      'Tiempo de conmutacion  electrovalvulas (segundos)\n'\
776   'Electrovalvula incorrecta, valor entre 1 y 4\n')
      return 0
778
    pico = float(sys.argv[6])
780 if pico<0:
      print ('VALOR MAXIMO INCORRECTO.\n' \
782   'Los parametros que se deben pasar son:\n'\
      'Succion del motor (50% -100%)\n'
784   'Temperatura inicial TGS2600 (50%- 100%)\n'\
      'Tiempo de experimentacion (minutos)\n'\
786   'Tiempo de conmutacion  electrovalvulas (segundos)\n'\
      'Electrovalvula incorrecta, valor entre 1 y 4\n')
788   return 0

790 bajo = float(sys.argv[7])
    if bajo<0:
792   print ('VALOR MAXIMO INCORRECTO.\n' \
      'Los parametros que se deben pasar son:\n'\
794   'Succion del motor (50% -100%)\n'
      'Temperatura inicial TGS2600 (50%- 100%)\n'\
796   'Tiempo de experimentacion (minutos)\n'\
      'Tiempo de conmutacion  electrovalvulas (segundos)\n'\
798   'Electrovalvula incorrecta, valor entre 1 y 4\n')
      return 0
800


802
    ## Vector que contiene el orden de conmutacion de las electrovalvulas
804 vectorValvulas = [int(odorante)]#
      #[1,1,2,2,3,3,4,4,5,5,4,3,2,1,3,1,4,2,4,1,5,2,5,3,5,1]##[1,2,3]##[int(odorante
      )]##
```

```
    ##numpy.random.randint(1,5,int((tiempo*60)/cambio)) ##[int(odorante)] ## numpy.
      random.randint(1,5,int((tiempo*60)/cambio))
806
    ##vectorValvulas =[4,4,4]
808
    vectorT= range(0,100,5)
810 vectorTd= range(95,-5,-5)

812 ## Se configura los puertos ADC, PWM
    motorStart(succion)
814 ADC.setup()
    PWM.start(heatPin2600,heat2600)
816


818 samples = 0
    while samples < 30:
820   tick = time.time()
      print ('Medidas de estabilizacion:\t'+str(readADC())) ## primera medida
      erronea
822   samples+=1
      tack = time.time();
824   time.sleep(SLEEP-(tack-tick))

826 ## precalentamiento del sensor
    calentamientoPrevio(vectorT,vectorTd,int(odorante))
828
    ## Generar Target
830 generarTarget(heat2600, periodo,pico,bajo)
    print(str(heat2600))
832 PWM.set_duty_cycle(heatPin2600, heat2600);
    esperar(3)
834
    ## Llamada al hilo para la medida de temperatura y humedad
836 ##_thread.start_new_thread(sensorTyH,(tiempo,))

838 ## creacion archivo de informacion experimento
    fileInfoTGS2600(vectorValvulas, succion, heat2600, tiempo, target,Kp,Ki,Kd,
      numCiclos,periodo)
840
    ## Sistema de control
842 contador = 1
    muestra = 0;
844 pos = 0 ;   ## cambiar este valor solo es una prueba "aire"
    temp = heat2600;
846 flagTarget = False
    temPIDmax =0.0
848 temPIDmin =0.0
    """print(str(pid.Kp)+"\t"+str(pid.Ki)+"\t"+str(pid.Kd)+"\n"\
850 +str(pid.sample_time)+"\t"+str(pid.current_time)+"\t"+str(pid.last_time)+"\n"\
    +str(pid.last_error)+"\n"
852 +str(pid.PTerm)+"\t"+str(pid.ITerm)+"\t"+str(pid.DTerm));
    tempActual = heat2600"""
854 lastTime = time.time()
    while contador < tiempo*60:
856
      electrovalvula = vectorValvulas[pos]
858   if electrovalvula == 1:
        print ('ELECTROVALVULA: '+str(electrovalvula)+' METANOL\n')
860   elif electrovalvula == 2:
        print ('ELECTROVALVULA: '+str(electrovalvula)+' ETANOL \n')
862   elif electrovalvula == 3:
        print ('ELECTROVALVULA: '+str(electrovalvula)+' BUTANOL \n')
864   else:
```

```
        print ('ELECTROVALVULA: '+str(electrovalvula)+' AIRE \n')
866     abrirElectrovalvula(electrovalvula)

868     j = 0;
        while j < cambio:
870       tick = time.time()
          if muestra == 0:
872         subtarget = target[muestra]
            muestra+=1
874         flagTarget = False
          elif (muestra%(periodo-1) ==0):
876         subtarget = target[periodo-1]
            muestra = 0
878         flagTarget = True
          else:
880         subtarget = target[muestra]
            muestra+=1
882         flagTarget = False

884


886

          datos = controlSistema(contador, subtarget,muestra,lastError,lastTime,
        addError,temp,flagTarget)
888       print ("temPIDmax:" +str(temPIDmax)+"\t temPIDmin:"+str(temPIDmin)+"\n")
          lastTime = datos[3]
890       addError = datos[2]
          lastError = datos[1]
892       temp = datos[4]
          contador+=1
894       j+=1
          tack = time.time();
896       time.sleep(SLEEP-(tack-tick))
        pos+=1
898
    finExperimento()
900 return 0

902 if __name__ == '__main__':
    try:
904   main()
    except KeyboardInterrupt:
906   print ('Interrupted')
      finExperimento()
908   sys.exit(0)
```

resources/principalPID_placa2_v1.py